The MagPi

Issue 103 | March 2021 | magpi.cc

The official Raspberry Pi magazine

Easy Pico Projects

Turbocharge
Raspberry Pi 400
with an SSD drive

#MonthOf
MAKING

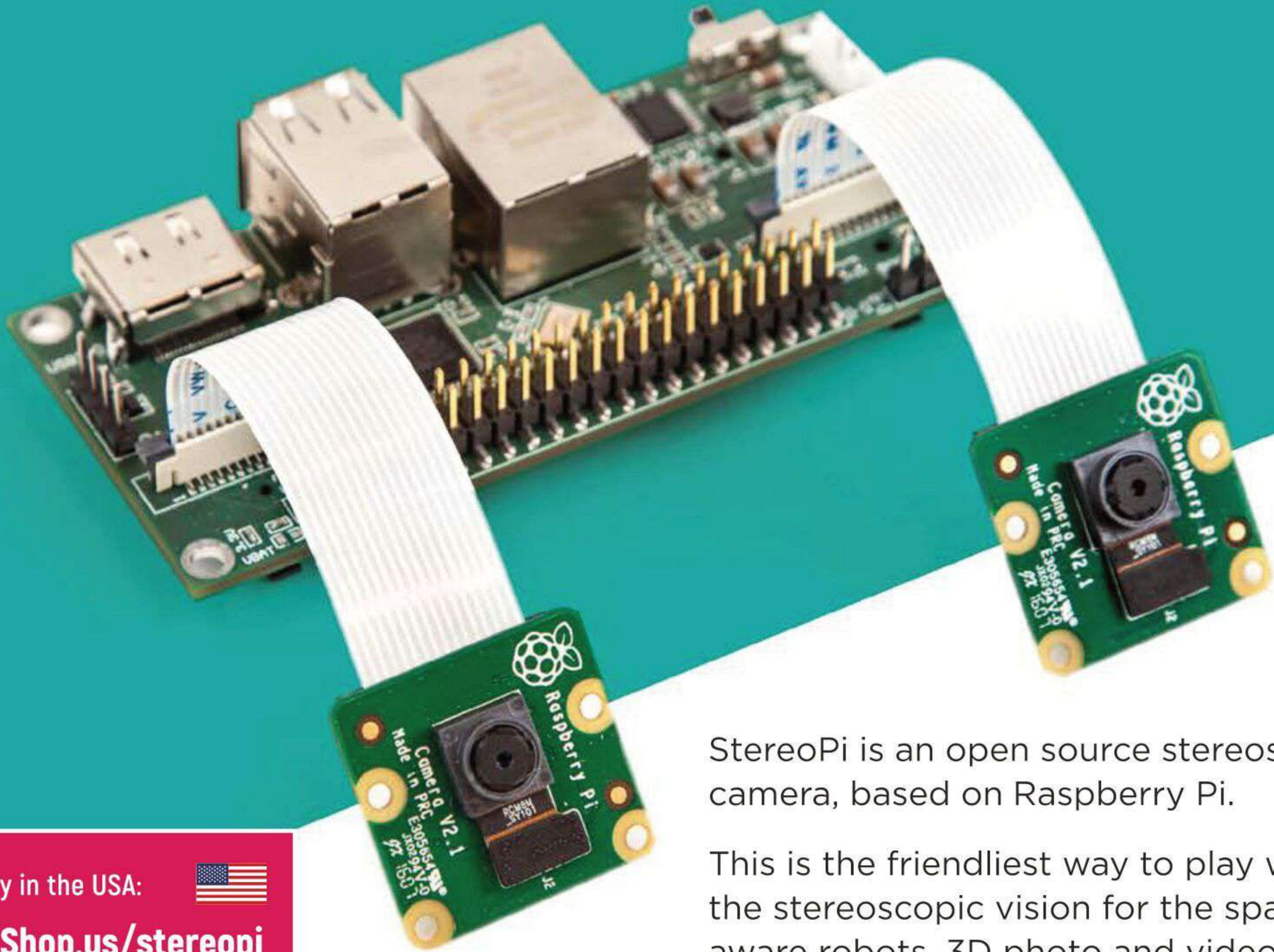Discover new ways of making
with Raspberry Pi

Upcycle an
Amiga 600

GLOBAL
DELIVERY
magpi.cc/store

Raspberry Pi
PRESS

Build an on-air
meeting light

42 PAGES OF PROJECTS & TUTORIALS

WELCOME
to *The MagPi* 103

Spring is in the air. The British sun has got–his–hat–on at a jaunty angle and we're looking forward to a summer of sunshine and outdoor activity.

This issue we're on our third ever #MonthOfMaking. Every March we throw off winter, dust off our tools, and encourage everyone to get making. It doesn't matter what you plan: from the smallest mouse of a make to that massive computerised elephant of a project you've always dreamed of. Now is the time to get started.

Rob has a great feature for #MonthOfMaking (page 30) that covers a range of crafting techniques. Discover a range of new materials to bring to your build. Whether you're into woodworking, making with metal, testing out textiles, or sticking to classics like plastic, cotton and wool, there are tips for everybody.

Of course, this being *The MagPi*, we like to feature Raspberry Pi at the heart of our builds. Last month we introduced Pico, the tiny microcontroller from Raspberry Pi. This month we're looking at Easy Pico Projects (page 64). We're having a huge amount of fun testing Pico and thinking up all the things we can make with it.

Don't forget to share your #MonthOfMaking projects with the rest of the community. We want to see what you build. Happy #MonthOfMaking!

Lucy Hattersley Editor

GET A
RASPBERRY PI ZERO W KIT
WITH A SUBSCRIPTION!
PAGE **28**

Contents

> Issue 103 > March 2021

30

20

Bird and Squirrel Detector

Raspberry Pi Amiga 600

Tutorials

The Big Feature

Pico Pomodoro Timer

Raspberry Pi 400
with SSD drive

Easy Pico Projects

SmartiPi Touch Pro

Reviews

Community

Tanya Fish interview

WIN 1 OF 10 **RFID** HATS! **95**

Learn at Home programme continuing to provide computers to students

Disadvantaged young people are being helped by the Raspberry Pi Foundation while they study at home. By **Rob Zwetsloot**

The running of schools right now is in flux. What this does mean, though, is that many more children are now learning from home than they were last year, and may be for the rest of the school year.

While some families may have the resources to allow for their children to continue schooling from home, some may not.

"The closure of schools has called attention to the digital divide, which sees poorer families struggling or unable to access education," the Raspberry Pi Foundation tells us. "The coronavirus pandemic didn't cause this divide, but it has highlighted it and its impact on many people in our society. This has become significantly more urgent as a result of the pandemic and the ongoing disruption to schooling, which we know has a disproportionately negative impact on children who already experience disadvantage."

Stay connected to school

Ofcom estimates about two million children in the UK do not have access to a computer for learning from home. Some charities, and even the Department of Education, have stepped up to allow for loaning laptops to students in need, and the Raspberry Pi Foundation is no different, donating computer kits to nearly 4000 young people.

> ❝ The Foundation's staff are also made available for ongoing tech support ❞

"Since April 2020, we have partnered with a network of over 40 schools and youth and community organisations across the UK to get computers into the hands of nearly 4000 disadvantaged young people that lack a computer to continue their school work," the Foundation explains. "This work was generously funded by the Bloomfield Trust, S&P Global Foundation and more than 70 other generous individuals and companies."

Recipients are some of the more poorer families that don't meet the requirement for the Department of Education's loan initiative. These families also get support to get the computer

Stay Connected kit

The whole thing costs under £200 – here's what students are getting:

- Raspberry Pi 400
- Mouse
- HDMI cable
- Monitor
- USB webcam with microphone
- Headphones
- Power supplies

Where the internet isn't available, the Raspberry Pi Foundation is helping to provide low-cost or free internet access, where possible.

set up, and the Foundation's staff are also made available for ongoing tech support.

▲ Computer kits are sent to the recipients as part of the scheme

▼ Help is available for setting up the computer with all the items in the pack

Immediate impact

For a program of this scale, the Foundation has made sure to get feedback where it can – from surveys to feedback from partners – and so far it looks promising:

- Young people who previously weren't engaged have begun engaging with learning.
- Parents have reported positive changes in young people's attitudes and behaviour.
- Youth and social workers have deepened their relationship with families, enabling them to provide better overall support throughout this crisis.

The program is ongoing throughout 2021.

"Building on the success of our programme to date, we are now seeking more financial and practical support to help us achieve further impact at significant scale. If this is something you can support us with, please contact please contact us by visiting **magpi.cc/learnathome**." 🅜

METAR Map

A colour-coded weather map provides an at-a-glance insight into whether it's good enough weather for flying, learns **Rosie Hattersley**

MAKER

Philip Rueker

Philip is a software engineer at Microsoft's Redmond, Washington headquarters. This is the first Raspberry Pi project he's designed and built from scratch.

magpi.cc/ metarmap

Warning!
Weather warning

Weather conditions can change abruptly, so consult detailed local forecasts before setting off for sky-bound adventures.

magpi.cc/ aviationbriefing

▶ METAR data can be pulled from a site such as **aviationweather.gov**, which uses familiar airport short codes. Write these codes on the back of your map when attaching your LEDs

Learning to fly your own plane is an idle fantasy for many of us. Such heroic, escapist dreams can be triggered by the sight of a small craft passing overhead. So it was for Austrian-born, US-based software engineer Philip Rueker. When leisure flights became a regular sight in his adoptive home of Redmond, Washington, Philip was thrilled enough to build his own aeroplane tracking tool. He's now built a Raspberry Pi Zero-based METAR (meteorological aerodrome report) map on which colour-coded LEDs show the current flying conditions at local airports with a weather reporting station. Before flying an aeroplane, "you have to do some planning for the weather. Having a map on the wall to get a first glance indication of whether the weather is good today is a great start," he says.

Philip had long been fascinated "that a small, tiny Raspberry Pi is more powerful than the first full-size computer I sat in front of a long time ago when I was a child." He also knew that Raspberry Pi would be very easy to use and set up, and expand its functionality over time. He decided on a Raspberry Pi Zero W since the program does not need a lot of power. He could connect to it over a wireless network to make changes to the code without having to plug it into the computer. He's recently added a mini LED display to the setup.

Plane spotting

Philip's previous projects include a PiAware aeroplane tracker which logs flights over his house and reports them to Flightradar24, along with a Raspberry Pi 3-based Stratux box which monitors nearby planes while you're in the air. These gave him a great start when designing the METAR Map, for which he was mainly focused on developing his Python skills.

Having seen the concept floated in a Reddit post, Philip and his partner – also a dedicated crafter and plane nerd – decided to work on it together. "It took
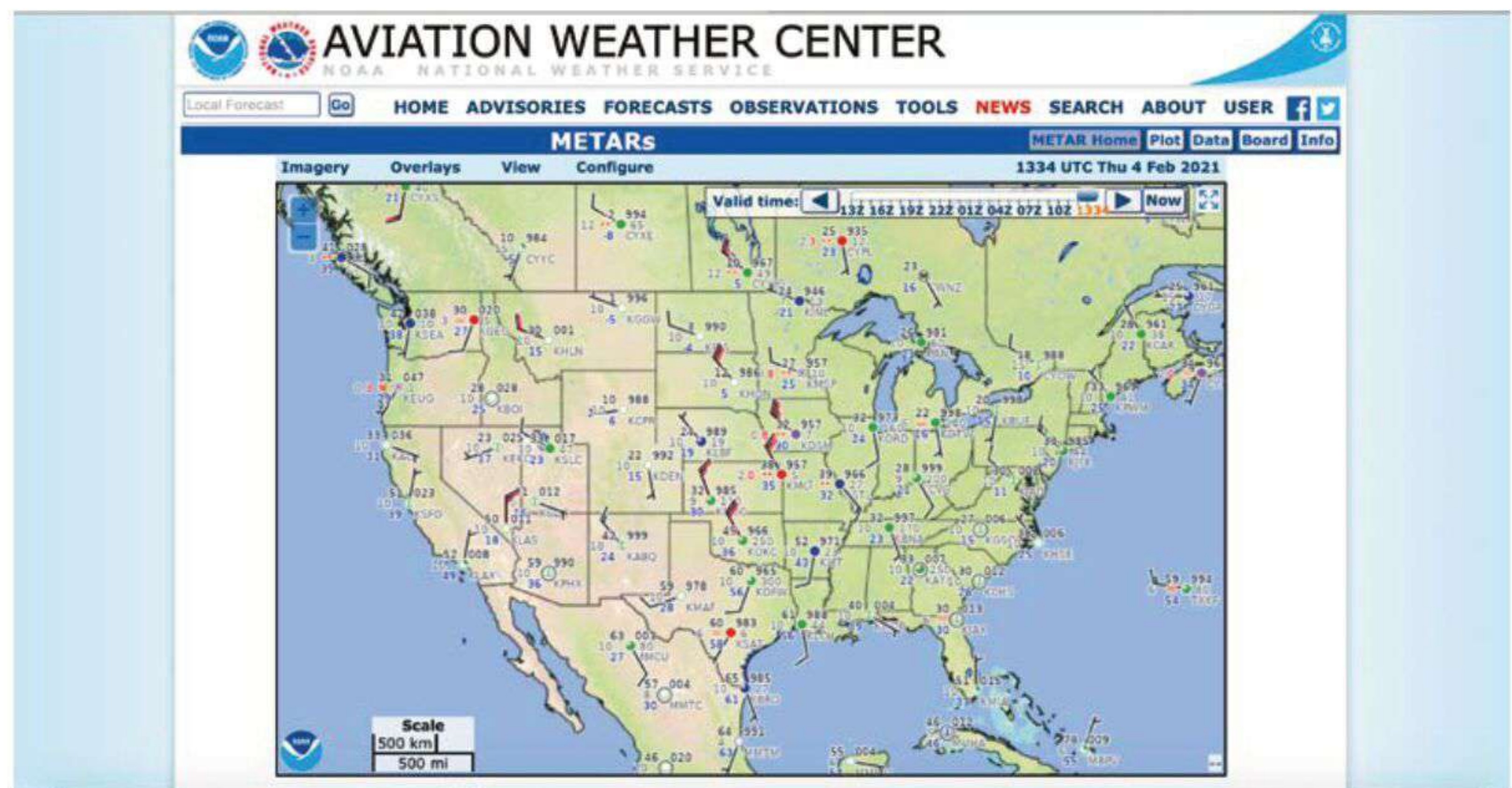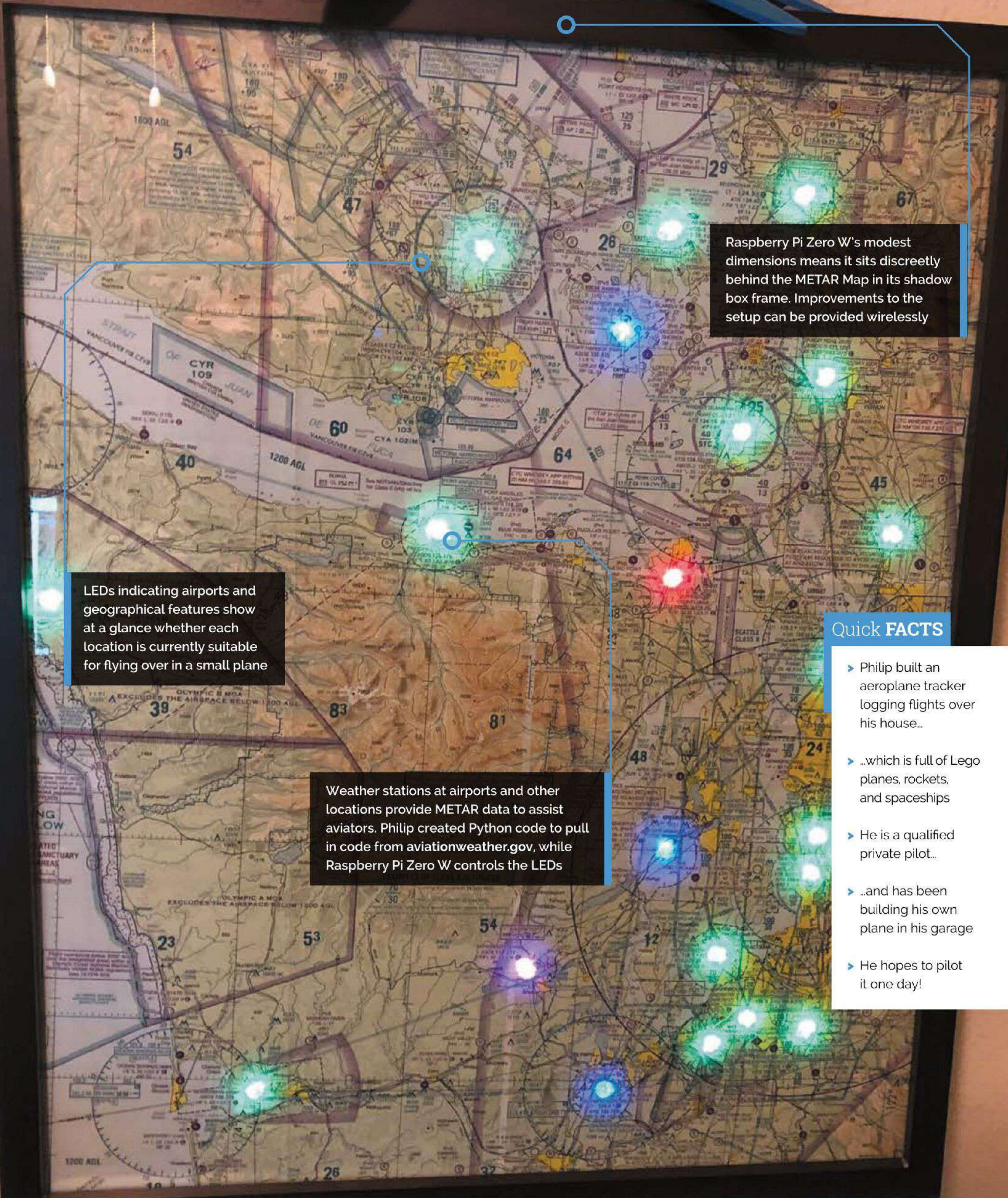
Raspberry Pi Zero W's modest dimensions means it sits discreetly behind the METAR Map in its shadow box frame. Improvements to the setup can be provided wirelessly

LEDs indicating airports and geographical features show at a glance whether each location is currently suitable for flying over in a small plane

Weather stations at airports and other locations provide METAR data to assist aviators. Philip created Python code to pull in code from aviationweather.gov, while Raspberry Pi Zero W controls the LEDs

Quick FACTS

> Philip built an aeroplane tracker logging flights over his house...

> ...which is full of Lego planes, rockets, and spaceships

> He is a qualified private pilot...

> ...and has been building his own plane in his garage

> He hopes to pilot it one day!

▲ The addition of an LCD provides detailed airport weather information – useful for take-offs and landings

▼ Testing the code on a string of LEDs before taping them to back of the map
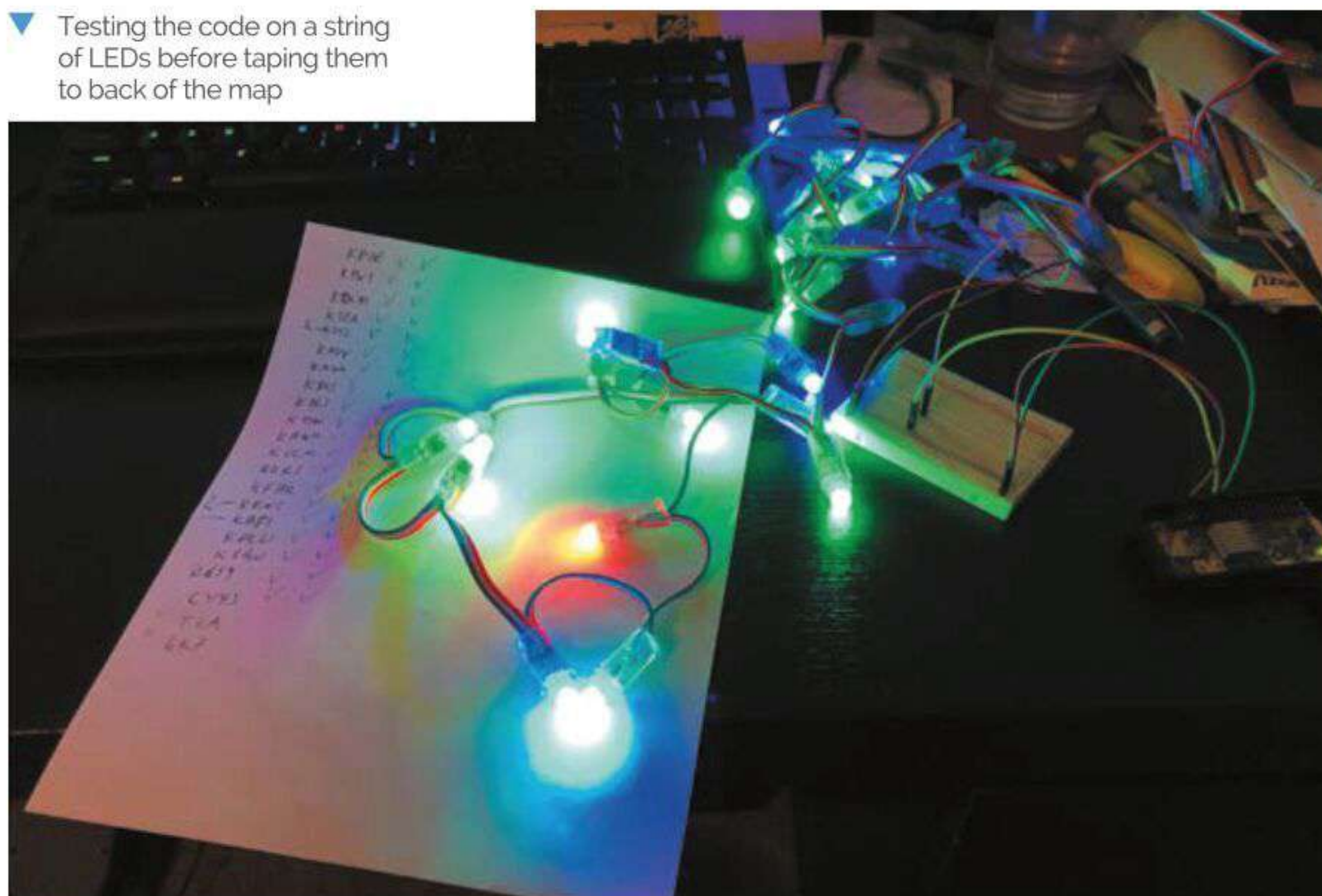
a little bit of learning about language and piecing together various libraries to make the lights, and everything come together," he says. The project cost around $100, with the shadow box picture frame representing the biggest outlay. It would be perfectly possible to make your own frame, of course. The LEDs and Raspberry Pi Zero require little power, so the running costs are negligible.

Philip wrote the code himself and is proud of the way he pieced the project together with eye-catching elements such as using the NeoPixel library to communicate with the LEDs, while keeping things simple so that others could build METAR maps of their own. Having posted the project on GitHub (**magpi.cc/metarmapgit**), Philip's been delighted by the "awesome" METAR maps other people have created and has added functionality based on GitHub users' requests.
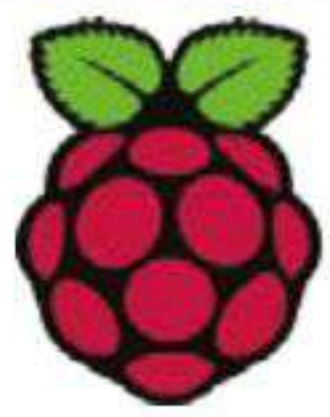
Raspberry Pi® Pico

Extension Family

Make Your Project Smarter With *Pico's Expansions DIY Kits*

Breadboard Kit

All in one Raspberry Pi Pico kit with onboard LEDs, button, buzzer & breadboard
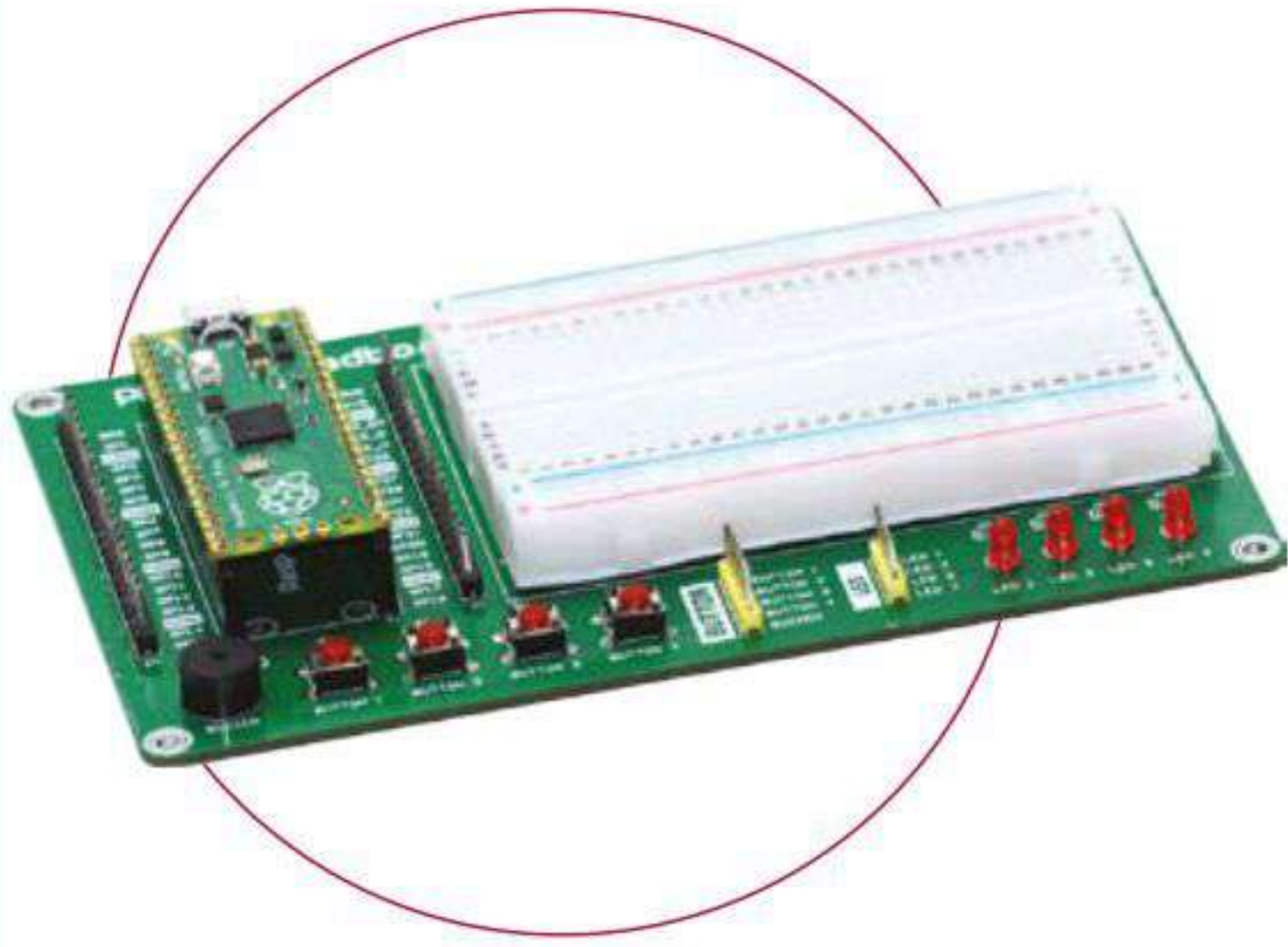
GPIO Expansion

Raspberry Pi Pico GPIO expander to access GPIO pins easily.
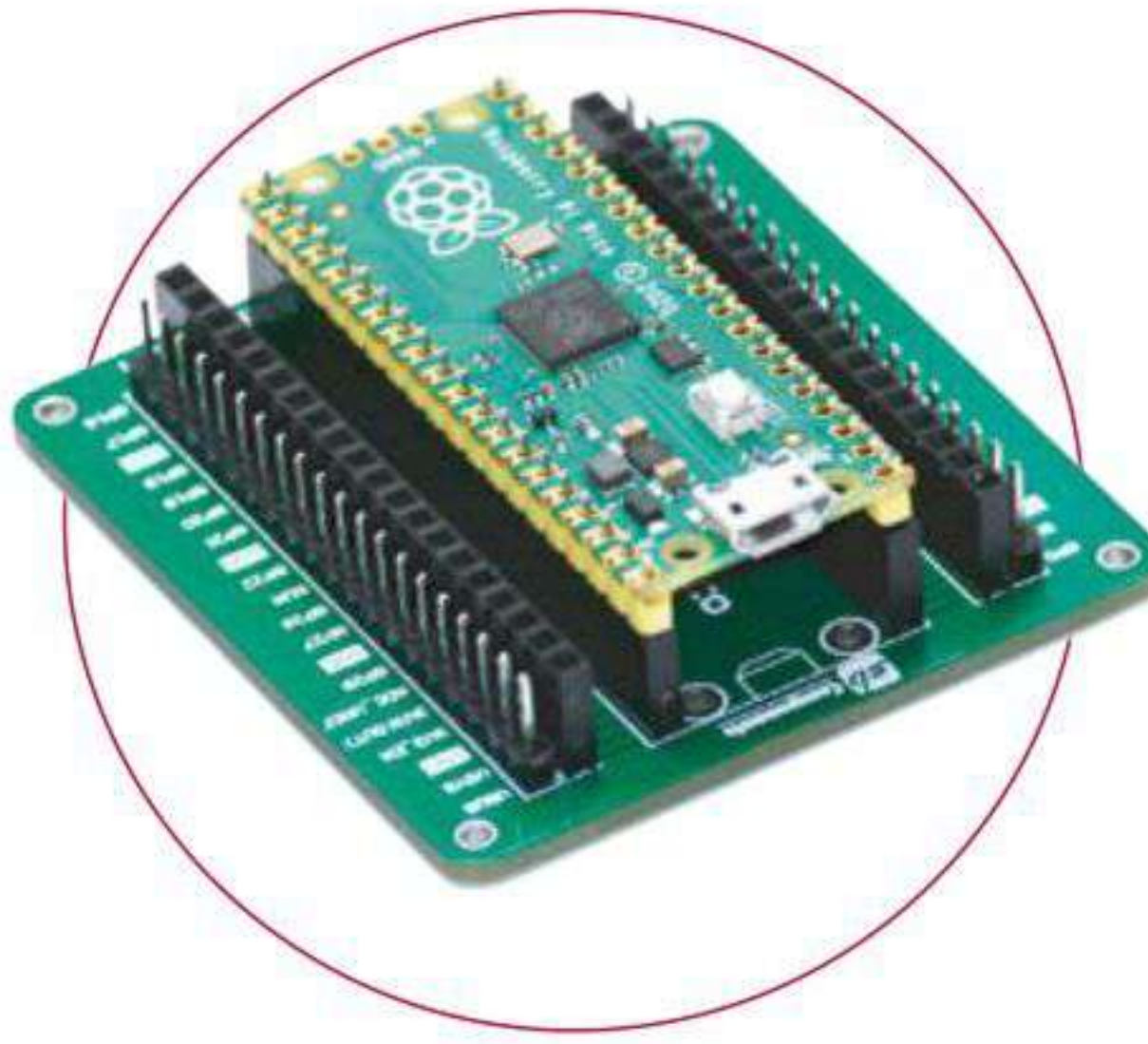
HAT Expansion

Connects any Raspberry Pi HAT with Raspberry Pi Pico.
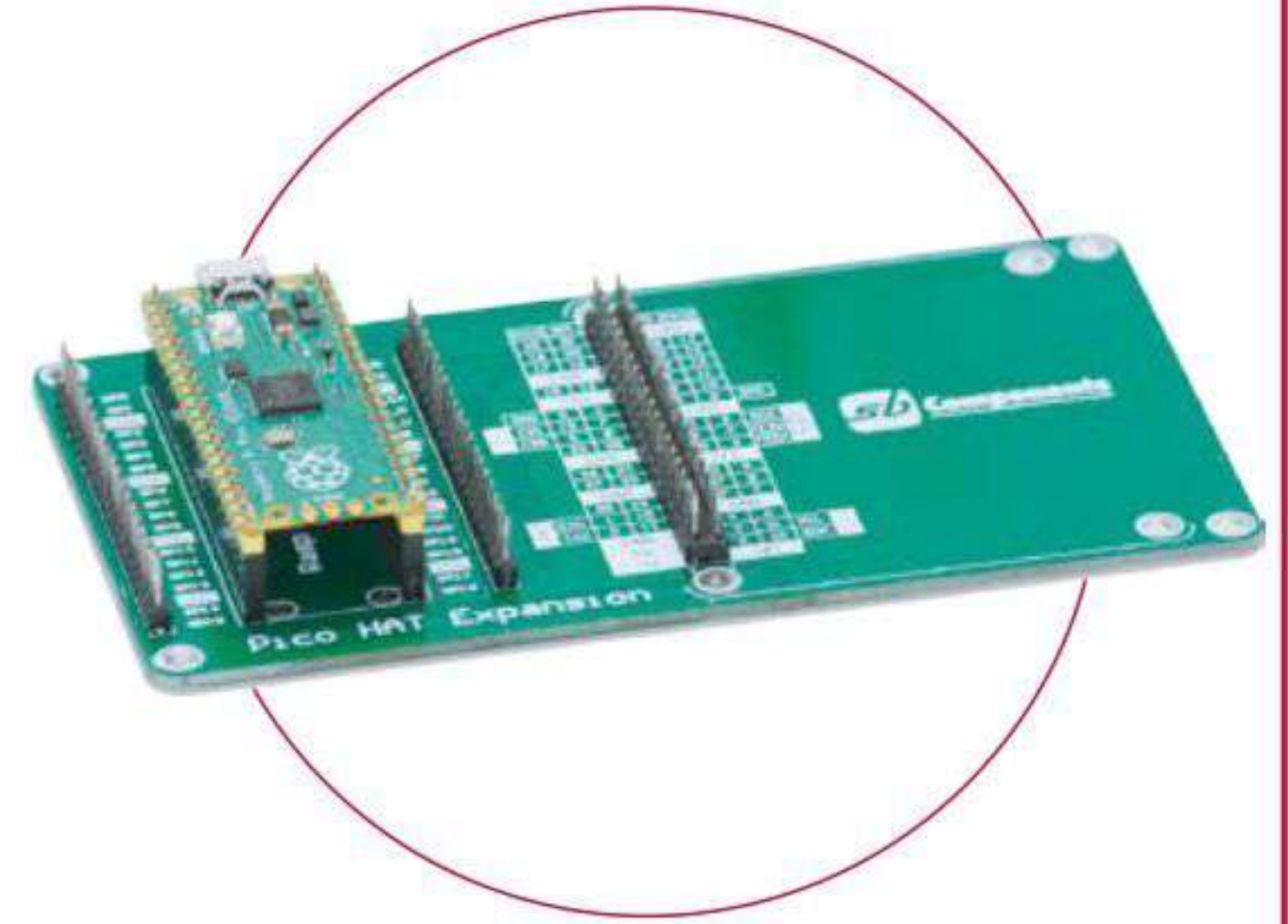
FOR MORE INFO, PLEASE VISIT : **SHOP.SB-COMPONENTS.CO.UK**
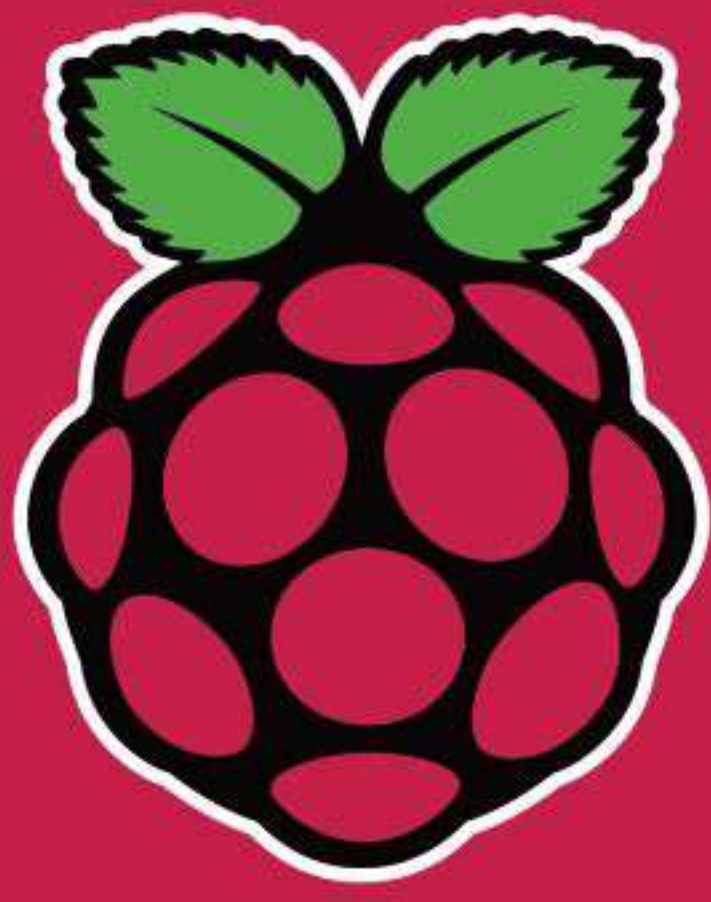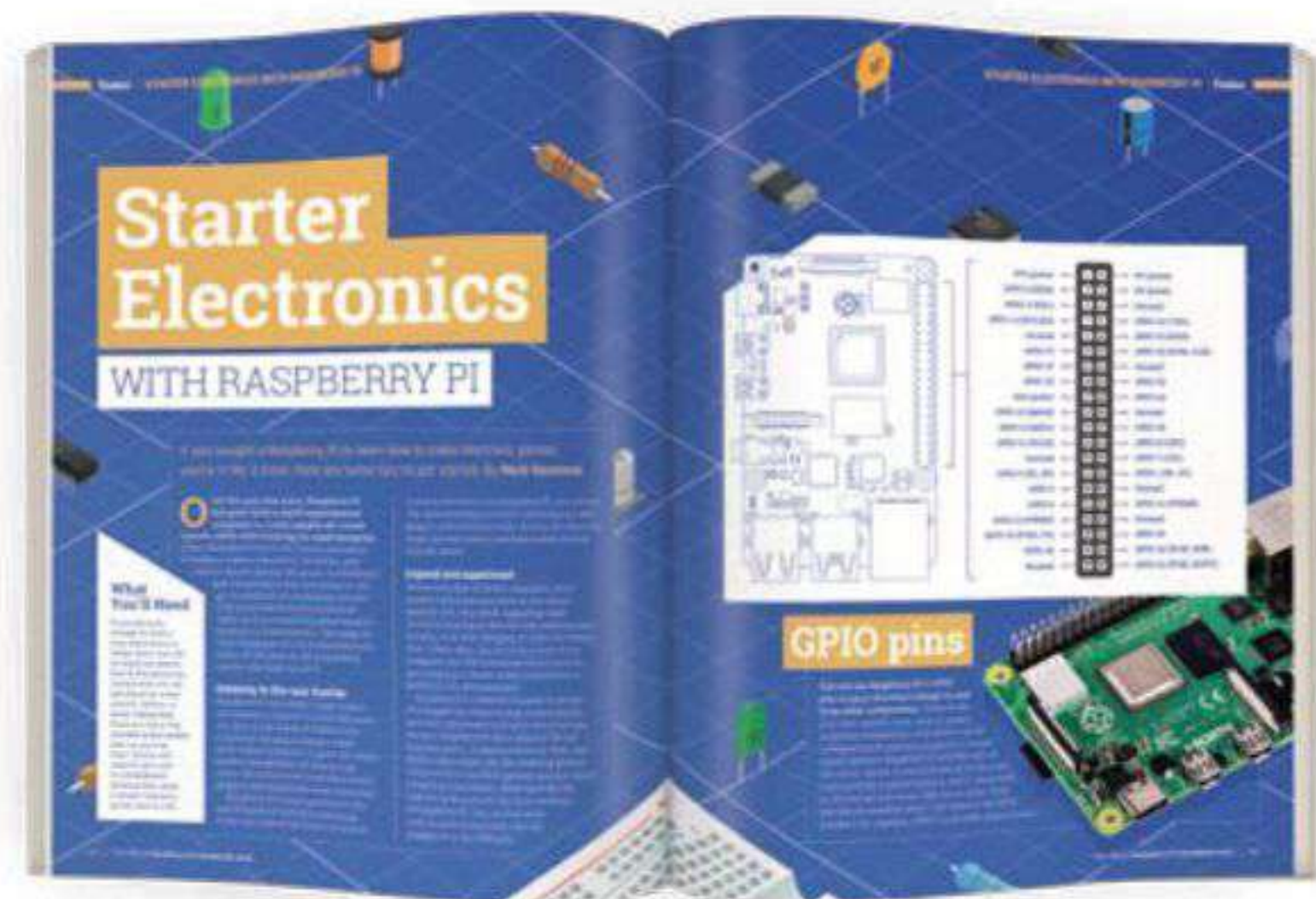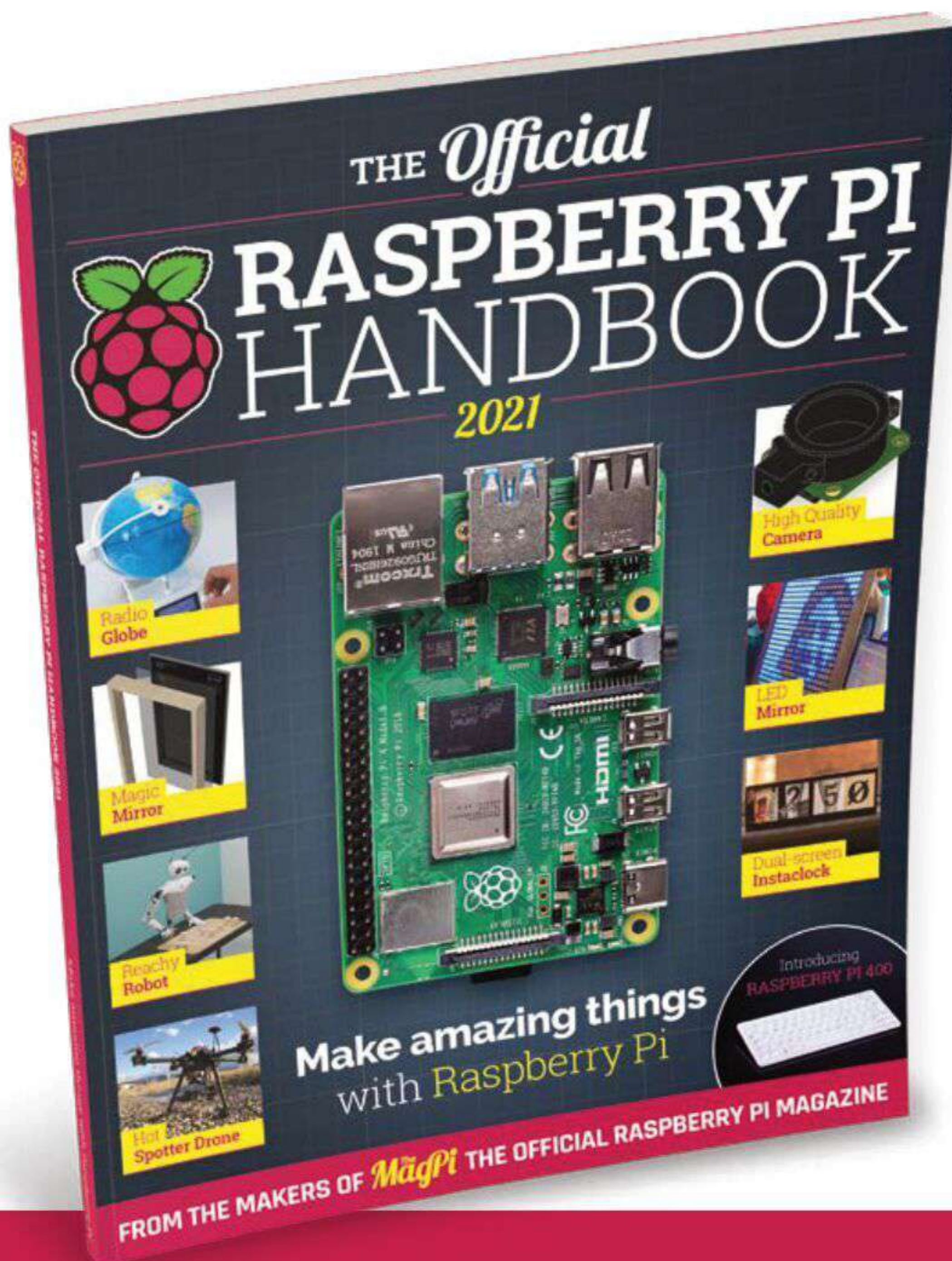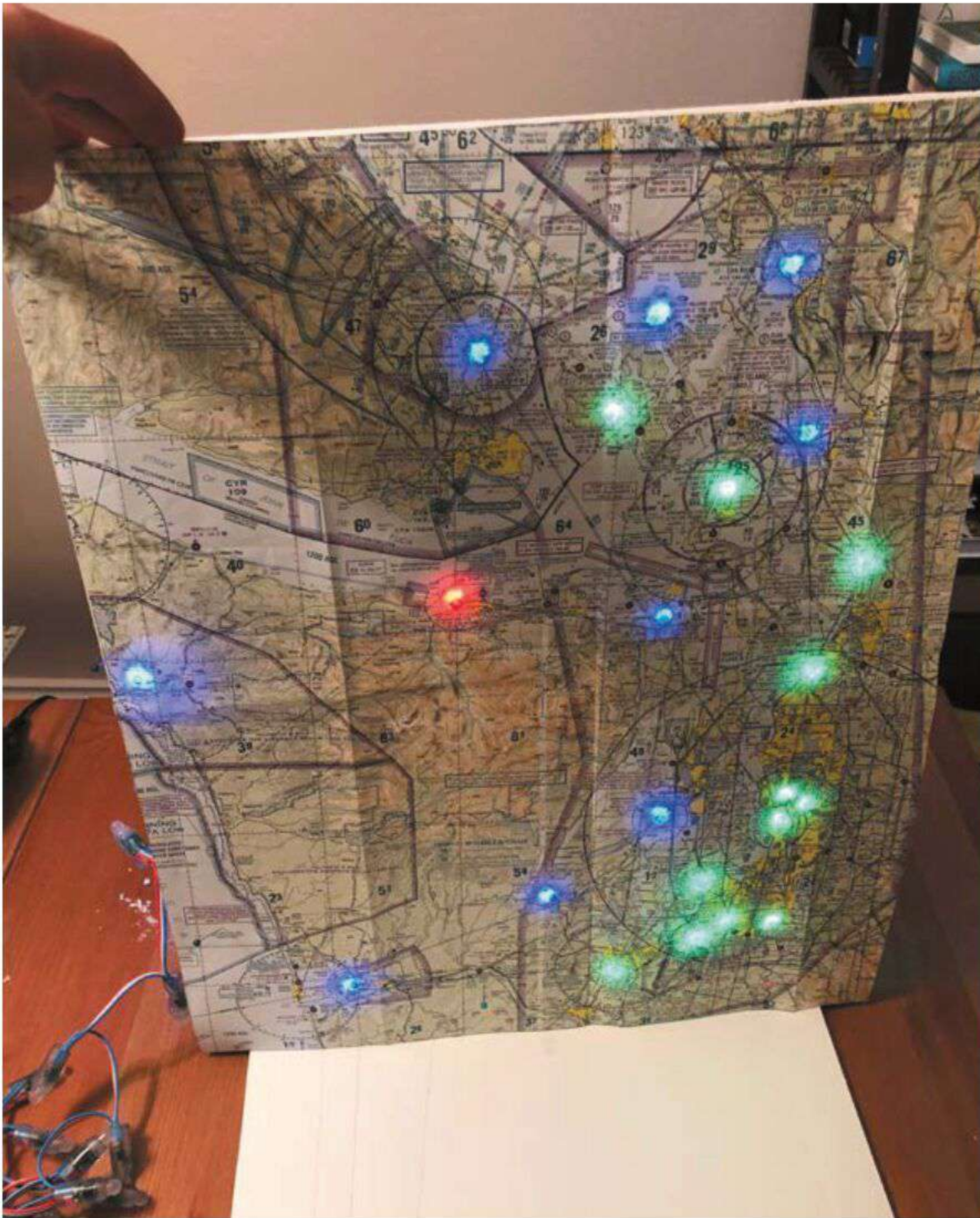
Refinements include making the lights blink if there are high winds at an airport, and another addition to make the LEDs flash white if there is a lightning storm in the area. He's recently added a small LCD which shows full weather information for the airports.

▲ In response to feedback by makers on GitHub, Philip added a lightning indicator and set LEDs to blink if a weather station is reporting high winds

> **"** Philip's been delighted by the awesome METAR maps other people have created **"**

Mighty maps

Although Philip created his METAR Map with aviation weather in mind, he says it could easily be adapted by someone who wanted to make a similar map to visualise the weather in nearby towns or cities. "All that would be needed would be an online source to get the weather data."

Fellow aviators have taken his project to heart. "I've had quite a few pilots contact me who said they have never written any code [but who] were able to successfully put it all together and showed me their creations." M

Make your own map

A great project for Python coders, you'll need NeoPixels or WS2811 addressable LEDs, a detailed map, soldering iron, and a Raspberry Pi Zero W. Find Philip's GitHub at **magpi.cc/metarmapgit**.
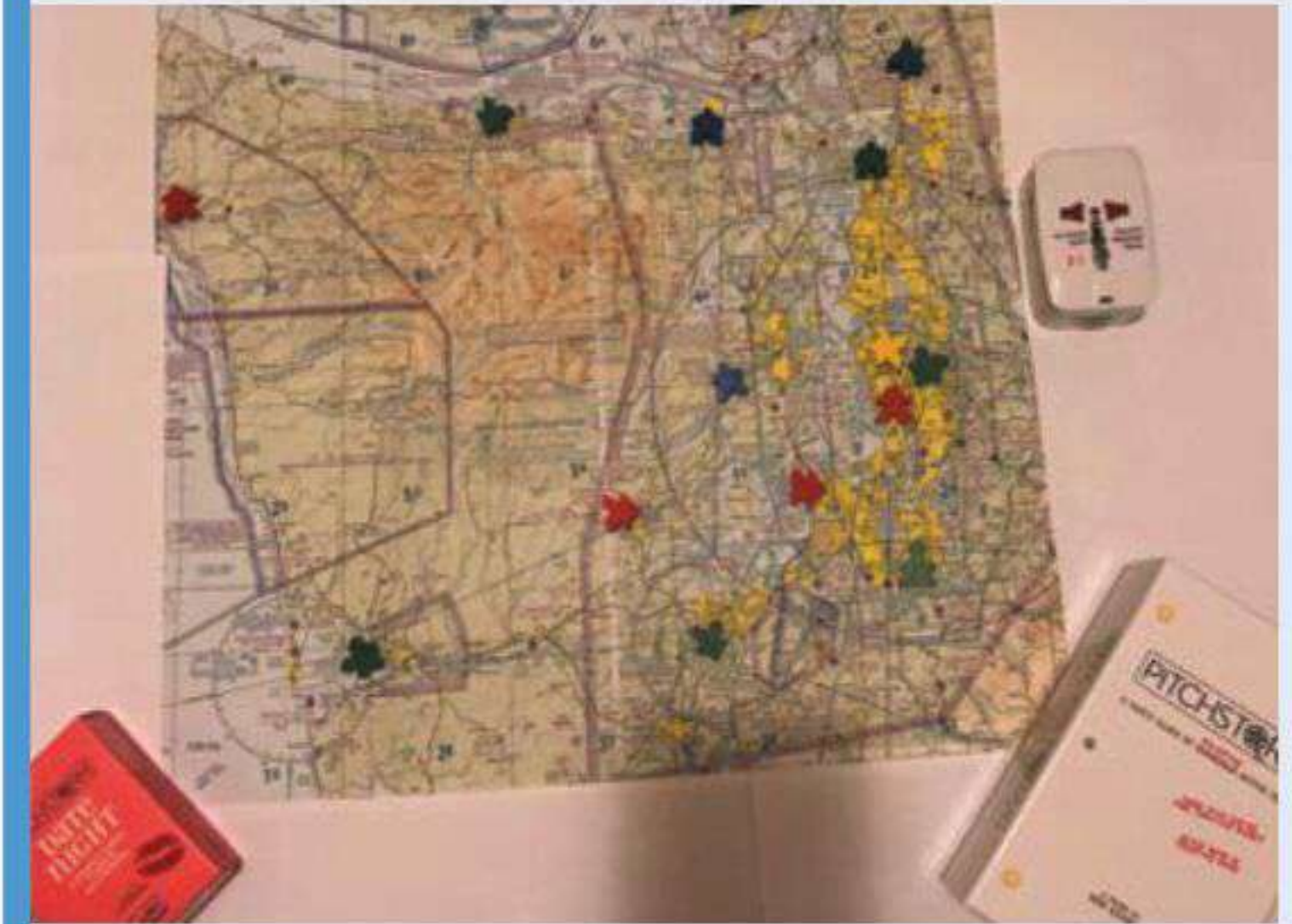
01 Lay out your map and identify cities, airports, and other locations you want to monitor. A comprehensive list of weather stations can be found at **magpi.cc/aviationweather**.

fritzing
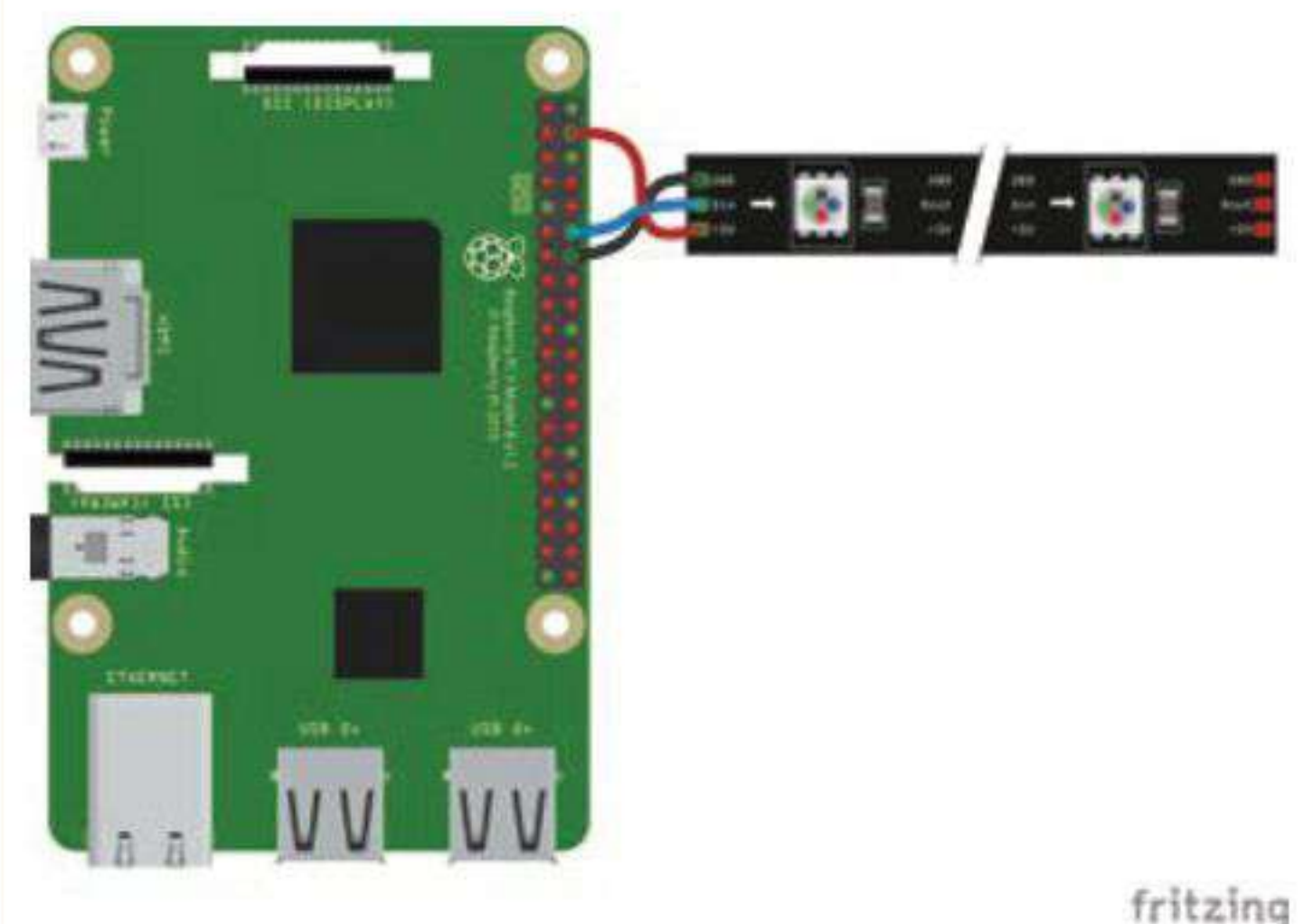
02 Attach LEDs to Raspberry Pi Zero W and use the code on GitHub to match up lights with your chosen METAR locations.

03 Test your lights, then tape everything to the back of your map (laminating or gluing it onto a board aids durability) and carefully place it in its frame.

AirMyPrayer

Upgrading prayer reminders with new technology was a perfect job for Raspberry Pi, as **Rob Zwetsloot** finds out

MAKER

Abid Shah

Abid has worked in IT for the last 21 years. His hobbies include reading, helping with charitable works, freestyle wrestling, and working on the AirMyPrayer project.

airmyprayer.co.uk

With broadband internet available to huge portions of the population, it's easier than ever to connect to people remotely. The ubiquity of video calls and conferences means that you can have online gatherings like never before. For Abid Shah, this meant opening up places of worship to more people.

"My project involves streaming live audio and video from houses of worship (actually from anywhere with internet) to social media platforms such as YouTube, Facebook and, more uniquely, straight to people's homes," Abid explains. "I have also designed and implemented an integrated prayer timetable."
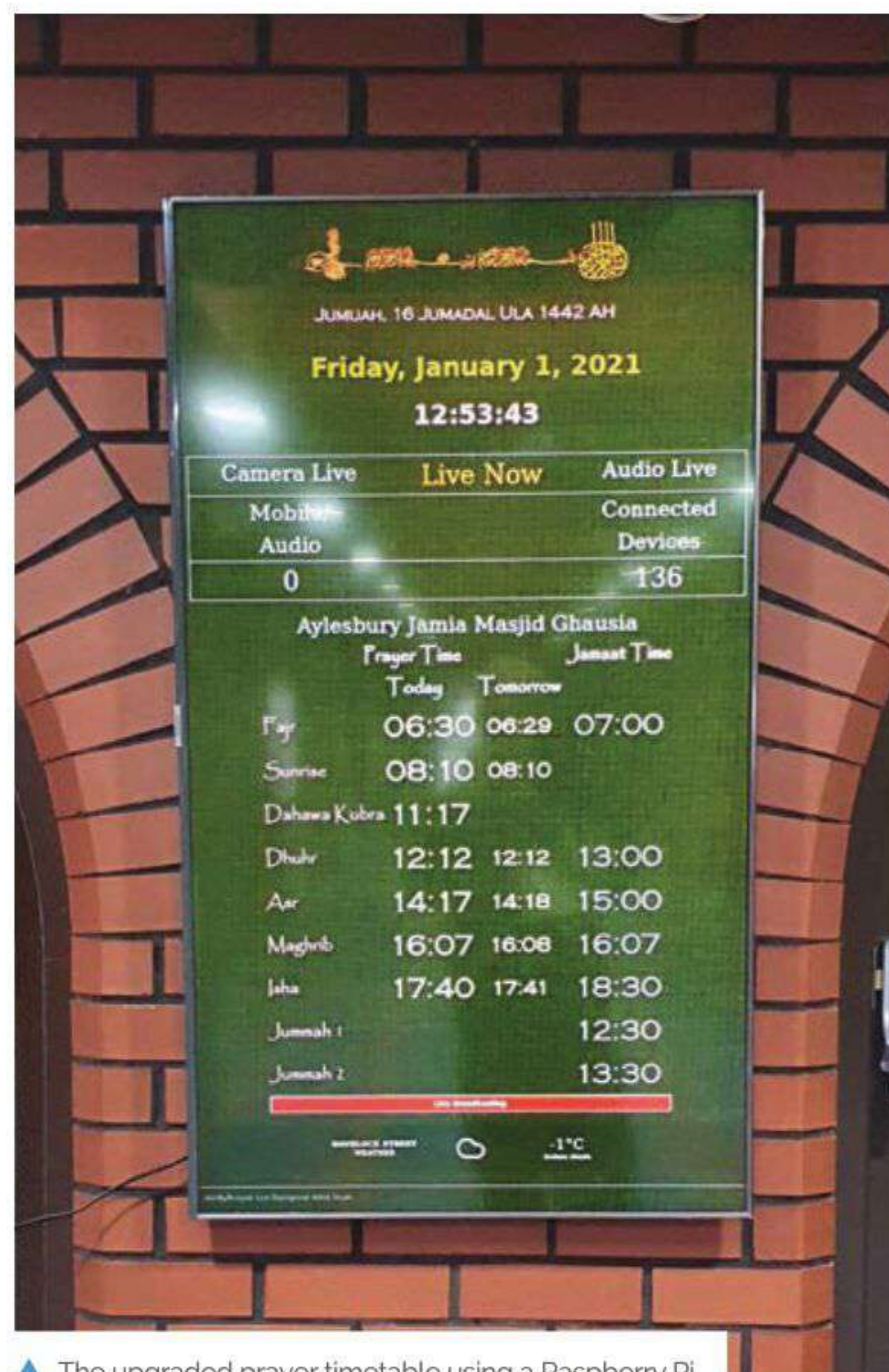
▲ The upgraded prayer timetable using a Raspberry Pi

The prayer timetable is something Abid has been working on for about nine years, when he noticed people were ringing up their local mosque to check on any changes to prayer times, which could happen every week. "This had me thinking that we need a way for the prayer times to be accessible on a virtual platform for users," he says.

Virtual timetable

Luckily, he was thinking about how to digitise the timetable at a very fortunate time. "After some research about what platform I could use to host such a project, the original Raspberry Pi was already on the market and seemed, in theory, to be the natural choice," Abid recalls. "Possibly the only practical choice as there was nothing else in the market in my budget range."

> **" As the internet is more accessible now than the range of UHF radio transceivers, it was a logical way of upgrading these calls to prayer "**

Using a client server setup, he was able to deliver a practical working example that is now being used in a several mosques. "To make it easier for the technophobes, I also have connected a Raspberry Pi to a smaller touchscreen monitor so one can easily change the congregational times," he adds.

The prayer timetable is only one part of the system – the other is a broadcasting system. "Mosques up and down the country traditionally have used UHF radio transceivers to transmit sermons or call to prayers to people's homes,"

You can look in on the mosque via a camera to feel more connected to the community

The prayer timetable is displayed on the screen and stays up-to-date

The home device can be plugged into a TV via HDMI
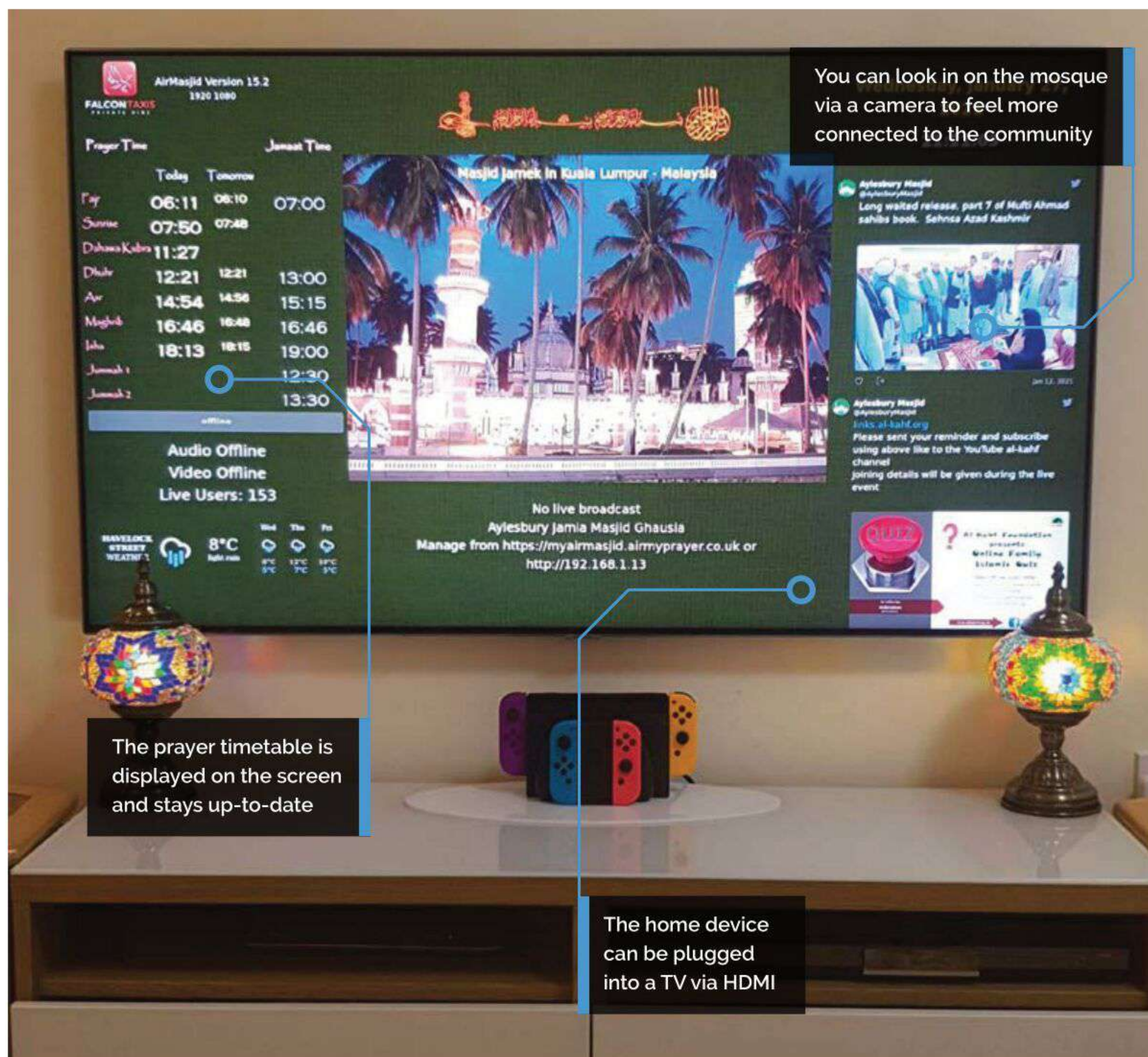
Abid says. "Unlike experiencing the call to prayer in Islamic countries over the loudspeaker, the best alternative was to receive it through UHF radio receivers installed in homes."

Online solution

As the internet is more accessible now than the range of these transceivers, it was a logical way of upgrading these calls to prayer. Abid got to work.

"I came up with three key requirements," he explains. The first is to deliver five times daily "a call to prayer and sermons/events to people's homes using audio and/or video reliably without any user invention and fully automated. Secondly, it needs to be a budget system as we're dealing with charitable organisations. Lastly, it needs to be portable so can be used in any organisation with internet availability."

The current AirMyPrayer system consists of a broadcasting Raspberry Pi at the mosque, which can use cameras or just a microphone, and a Raspberry Pi 4 that can receive the internet broadcast for people in their home. It uses a small touchscreen and is highly customisable – you can

even connect to it on a phone. Check the website for more details: **airmyprayer.co.uk**.

Reception has been mixed – the older system has been in use for a long time, so changing to a new one has not been quick, according to Abid. "However, with incremental improvements to the design and a focus on a more friendly user experience, the device became more accepted, and now there are over 150 devices around my local area and still growing." M

Quick **FACTS**

> A lot of trial-and-error and iteration occurred during the first two years

> Live streams can be accomplished with a phone camera as well

> Alerts can be sent to users from the 'managers' of a connection

> Prayer is held five times a day

> The default home screen page includes local info, such as a weather forecast

◄ A simple mosque-side AirMyPrayer setup, which allows for voice transmission

ML-based Bird and
Squirrel Detector

Want to distinguish a bullfinch from a buzzard in your garden, or whether squirrels are up to no good? Machine learning has the answer. **Nicola King** takes a walk on the wild side

MAKER

Mike Sadowski

Mike is an IT executive with a real-estate company in the New York area. Previously he was CTO at two venture-backed companies. On the side, he enjoys working on projects related to IoT and machine learning.

magpi.cc/ birdsquirrel

New York-based Mike Sadowski had been interested in machine learning (ML) for some time and wanted some original images that he could feed into an ML algorithm: **"I was interested in the challenge of trying ML with real data, not a canned data set,"** he explains.

One day, while looking out of his window, a flash of inspiration came to him. "You really want a lot of data for machine learning – the more the better. I was looking out the window at my bird feeder and I realised that there were probably hundreds of birds visiting it daily, so that would be perfect! I added squirrels to the mix because they are always hanging around the feeder, hoping they can figure out how to break in."

Eagle eye

And so, Mike began work on his Bird and Squirrel Detector, a marvellous make that utilises a Raspberry Pi, a High Quality Camera, some clever code, and Amazon Web Services image recognition (aka AWS Rekognition). Mike set his Raspberry Pi up to run PI-TIMOLO (**magpi.cc/pitimologit**), a downloadable software module that watches for motion and takes a snap when it detects any.

"I have a Python program that runs on my Raspberry Pi that watches a folder for new photos. If it sees one, it makes an API call to AWS to send the photo to an AWS 'bucket'," Mike tells us. In AWS, he has a Python Lambda function (a cost-effective way of running code) that watches the bucket, waiting for photos. His Lambda takes the photo that just arrived and sends it to Amazon Rekognition, which then uses its ML-based image recognition capabilities to try to assess what the photo contains.

> **❝ I was interested in the challenge of trying ML with real data, not a canned data set ❞**

"Amazon Rekognition replies with a list of 'labels' (that's a machine learning term that describes what an ML algorithm thinks is in the picture)," explains Mike. "Then my Lambda code looks at the labels and decides if the image contains a bird or squirrel. Based on this, it sends a message to an AWS service called Simple Notification Service (SNS). You can subscribe to an SNS 'topic' and ask it to send you emails or texts. So I have one SNS topic for birds and another for squirrels, so I know what's in each photo."

Winging it

Mike had to tweak some of software parameters in order that the trigger to take the photo was just how he needed it. He wanted images of the birds and squirrels and not anything else. "You want to make sure you don't miss good photos, but you don't want to snap a picture every time a tree branch moves in the background, or you'll end up with thousands of photos per day."

▼ Some of the many birds visiting Mike's garden that inspired his project

A High Quality Camera with a telephoto lens is pointed at the garden bird feeder

Running on a Raspberry Pi 4, PI-TIMOLO detects motion and then captures an image

The image is sent to the AWS Cloud, where Amazon Rekognition analyses it and generates a list of ID labels

▲ A red-bellied woodpecker pays a visit to Mike's bird feeder. Standard AWS Rekognition identified it as a 'woodpecker'

Quick FACTS

> The project took Mike several weeks (part-time) to complete

> He plans to add another Raspberry Pi to create some squirrel deterrents!

> Mike's GitHub code can be found here: **magpi.cc/ birdsquirrelgit**

> If you want to have a go, you'll need Python skills…

> …and a medium-level grasp of AWS

In addition, he says, "The other bit of fine-tuning that took some time was filtering out all of the uninteresting labels Amazon Rekognition returned. It tells you everything it thinks it sees in the picture. So it won't just identify animals, it will also tell you it sees a bird feeder, or a chair. Or it might tell you it sees trees and grass, which may be accurate, but you don't care about that." So, he built up a list of 'uninteresting' labels over time, and filtered them out so he was only informed of bird and squirrel sightings.

Mike describes the feedback he's had from other makers as "amazing", and is glad to share his insight into both the possibilities and limitations of AI. He's also discovered the fantastic Raspberry Pi team spirit: "A cool thing about the Raspberry Pi community is that you can reach out to people and they will really help you." M

▲ The camera is pointed at the bird feeder. Mike started with a cheap telephoto lens, but switched to one borrowed from his Canon EOS camera

Raspberry Pi Amiga 600

Billy Nesteroulis has created an Amiga computer
for the modern user, as **David Crookes** explains

MAKER

**Billy
Nesteroulis
(DJ Nest)**

Billy is an Amiga
musician and a
member of the
Vintage Computers
Society of Athens.
His team specialises
in 3D prints and he
loves to experiment
with Raspberry Pi.

magpi.cc/djnest

E ven though the Amiga range of computers ceased production in 1996 following a successful eleven-year run, many users remain determined to keep its memory alive.

Not only has a new magazine recently emerged (Amiga Addict), but Commodore's machine has resurfaced in various guises over the years. Its operating system, AmigaOS, continues to be updated, and there are even rumours that a revived A500 model will be released this year.

Such news excites Amiga fans. "The price of used Amigas has skyrocketed over the last five years and it's not an easy task to preserve an old computer," explains Billy Nesteroulis, aka DJ Nest. "If you own an old Amiga, it will eventually break: their electrolytic capacitors tend to leak. You'll need a new power supply, and some kind of memory expansion is ideal."

With a Raspberry Pi computer, however, such costs can be significantly lowered. As Billy has shown, it's possible to build an Amiga 600 from scratch with a Raspberry Pi 4 as the main unit.

▼ A nine-pin joystick from an original Amiga computer can be used with the USB adapter by Retronic Design (**retronicdesign.com**)
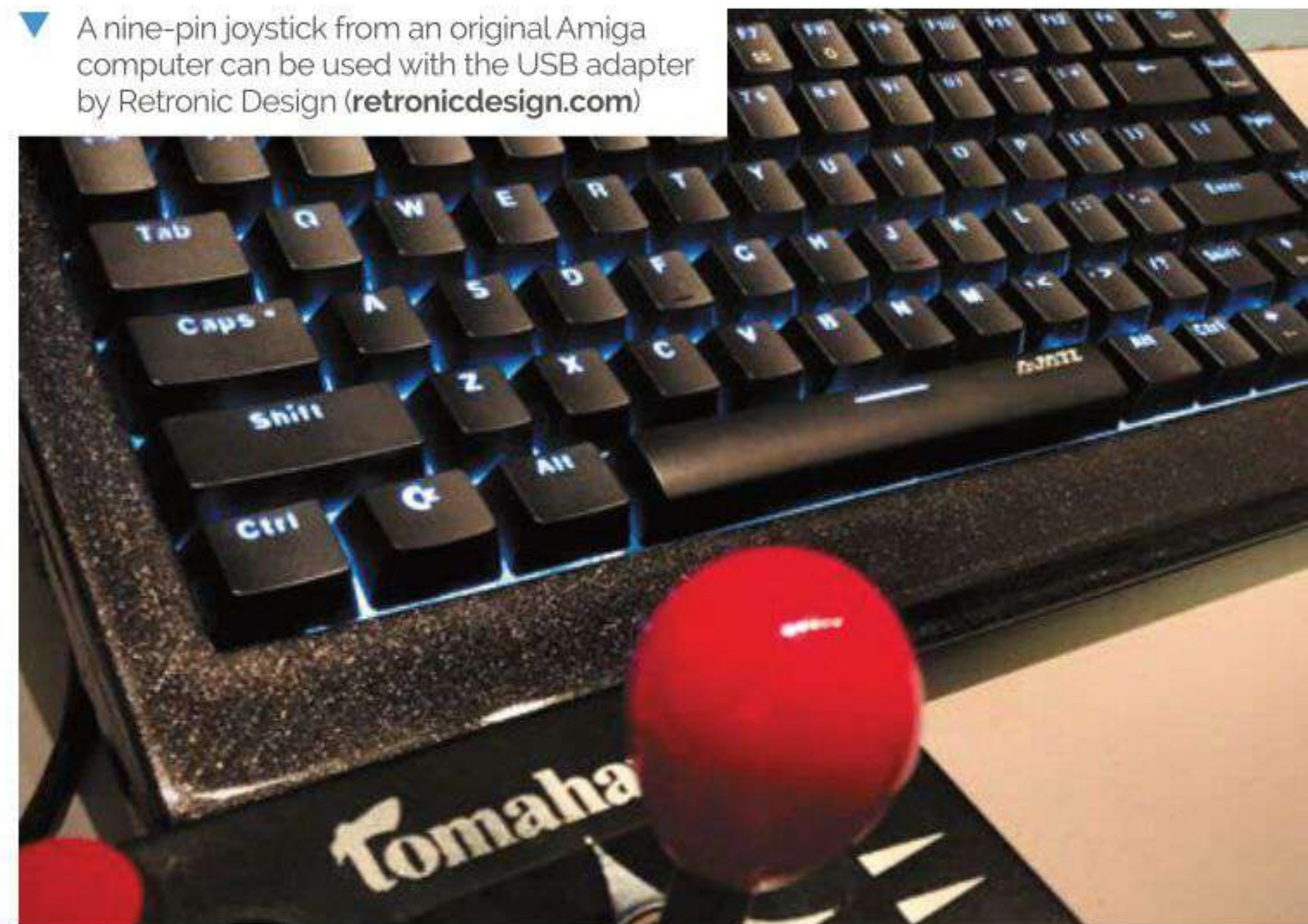
"Raspberry Pi can emulate an Amiga with AmigaOS and you can use it to play games and software made for the machine," he continues.

Stars in their eyes

Certainly, Raspberry Pi has proven to be the perfect platform for Amiga emulation. "Dimitris Panokostas has done a remarkable job creating the Amiberry emulator and because Raspberry Pi hardware is small, it can fit easily almost everywhere," Billy says.

In this instance, the single-board computer has been fitted inside a full-size, 3D-printed replica of an Amiga 600 case, allowing use of its USB ports and wireless LAN. A specially designed keyboard that was originally designed as a replacement for ageing Amiga machines is connected and modern adapters will allow use of the nine-pin joysticks of old for added authenticity.

"The Cherry MX keyboard is illuminated and it was designed to fit the case that I 3D-printed," Billy explains. "The joystick adapter is plug-and-play with no drivers needed and you can also use Amiga CD32 joypads with their eight buttons." Other parts include a micro HDMI extender, SD card extender, power supply unit, USB extenders, a power switch, and LAN extender.
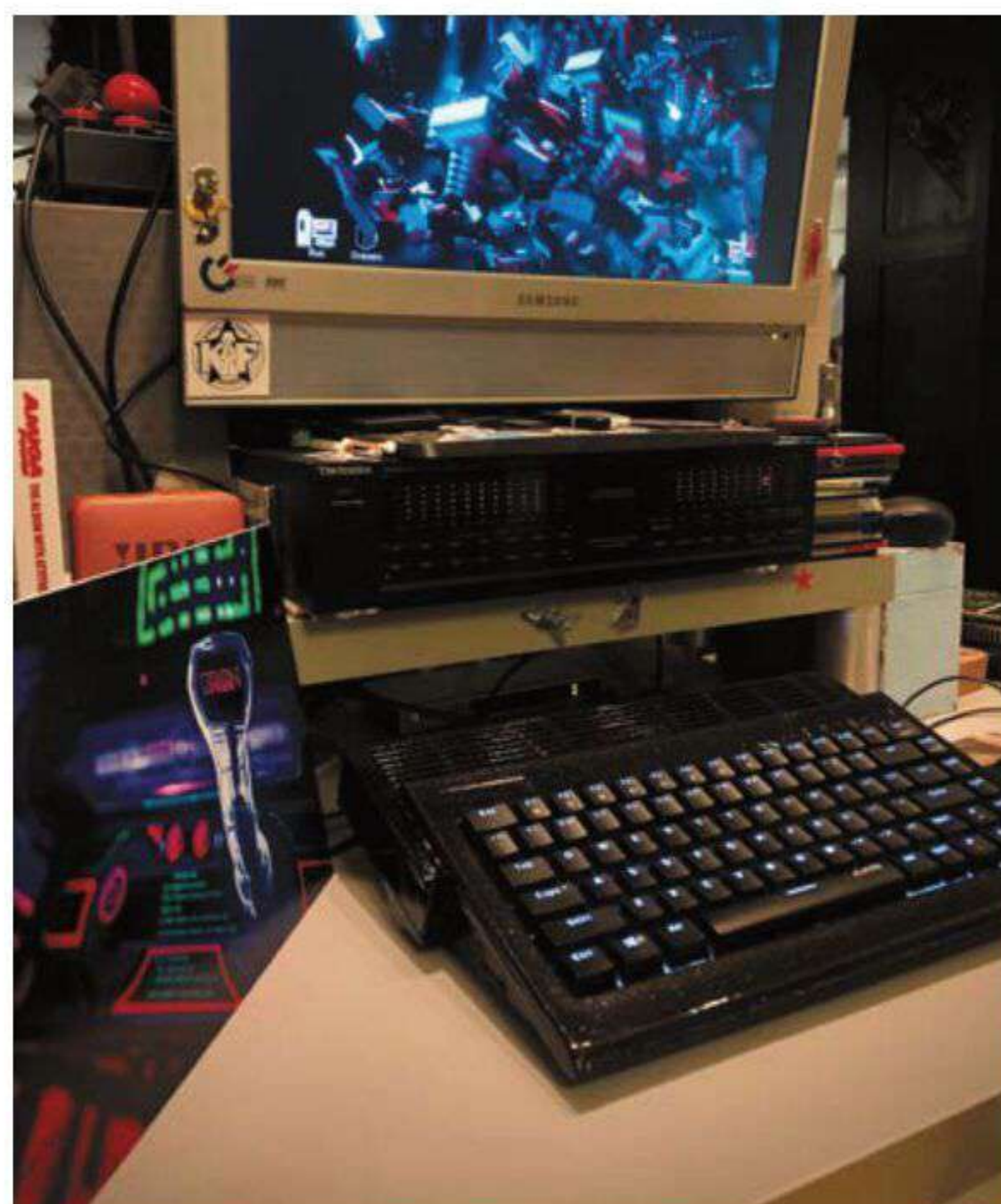
▶ With Amibian and Amiberry as the main emulator, you can emulate any Amiga model you like

Raspberry Pi has been overclocked from 1.5GHz to 2.1GHz, with the GPU running at 700MHz (up from 500MHz). A CPU heatsink with a fan is also used

The files for the 3D-printed case were created by Jens Mühlenberg and cost $20 to download from **magpi.cc/projjulia**

The Cherry MX keyboard is specific and designed to fit the case. You have a choice of a black or white keyboard with standard white lighting or RGB

Quick **FACTS**

> The project costs around $250 in total

> It can emulate Amiga 500s to Amiga 4000s

> But the case is modelled on an Amiga 600

> You can plug it into a modern monitor

> Amiga novices could use the PiMIGA emulator

To ensure everything runs smoothly, Billy uses the Amibian distro ("the most complete experience of the classic Amiga environment"). He also likes that – in exchange for a small donation – he can use the Amibian 1.5 Extended Edition made by Gunnar Kristjánsson. "The Extended Edition

> ## " Raspberry Pi can emulate an Amiga with AmigaOS and you can use it to play games and software "

includes Raspbian Buster V10 OS with the look and feel of the Amiga OS 4," Billy says. "It has a modern browser, the VLC media player, and the Qmmp audio player. You can even use LibreOffice Writer."

A modern touch

Amibian also allows users to update software and Amiga emulators through its configuration tool. All of which has meant Billy's set up expands the potential of the machine, beyond matching the real A600. "It's allowed me to bond classic computing with modern computing," he says.

Indeed, Raspberry Pi 600 gives the same feeling and experience of the A600, but with the modern touch of the Raspberry Pi hardware. "It has the required juice to run specific software such the classic pixel-art package Deluxe Paint, games play without issues, and you can build your own system and adapt it to your needs," Billy says. "For many people, it's the best Amiga solution in 2021." M

▲ Raspberry Pi Amiga 600 is the same size as the original A600

CNC plotter

Recycling old tech to allow a computer to draw? **Rob Zwetsloot** checks out this amazing machine

MAKER

Stratos Botsaris

A software engineer from Athens with an interest in DIY electronic projects using Raspberry Pi.

magpi.cc/ cncplotter

One of the reasons we highlight cool projects is because we hope they inspire others. Stratos Botsaris is one of those people that got inspired.

"I had seen people on the internet creating and using CNC plotters and always wondered how these machines work," he explains. "I was mainly interested in way the CNC machine translates the G-code instructions into movement that drives the stepper motors. I wanted to find out the internal workings of this."

That's just what he did, creating his own CNC plotter in the process. "It is controlled by Raspberry Pi and can draw an image on a surface the size of a piece of A4 paper," he tells us. "I have designed and built both the hardware and the software myself. I have assembled its hardware by using recycled parts from an old scanner and a printer."

He also wrote the Python software which runs on Raspberry Pi. "It is an interpreter which reads and executes the G-code from a text file and drives the stepper motors."

Switching to Raspberry Pi

According to Stratos, a lot of the projects he'd seen were made with Arduino, so he decided to see if it was possible with Raspberry Pi.

"I started experimenting with one stepper motor with a Raspberry Pi," he says. "Fortunately, I was lucky enough to have salvaged one stepper motor from an old printer and another one from an old scanner. In the beginning I had to find out how stepper motors work and how to connect one to Raspberry Pi. Then I tried to drive the stepper motor by writing a small program in Python and run it on Raspberry Pi. Once I managed to make this work, I got very excited and this gave me the push to continue with controlling two stepper motors at the same time. This was the most tricky part because I had to find a way to move the two stepper motors in parallel if I ever wanted the CNC
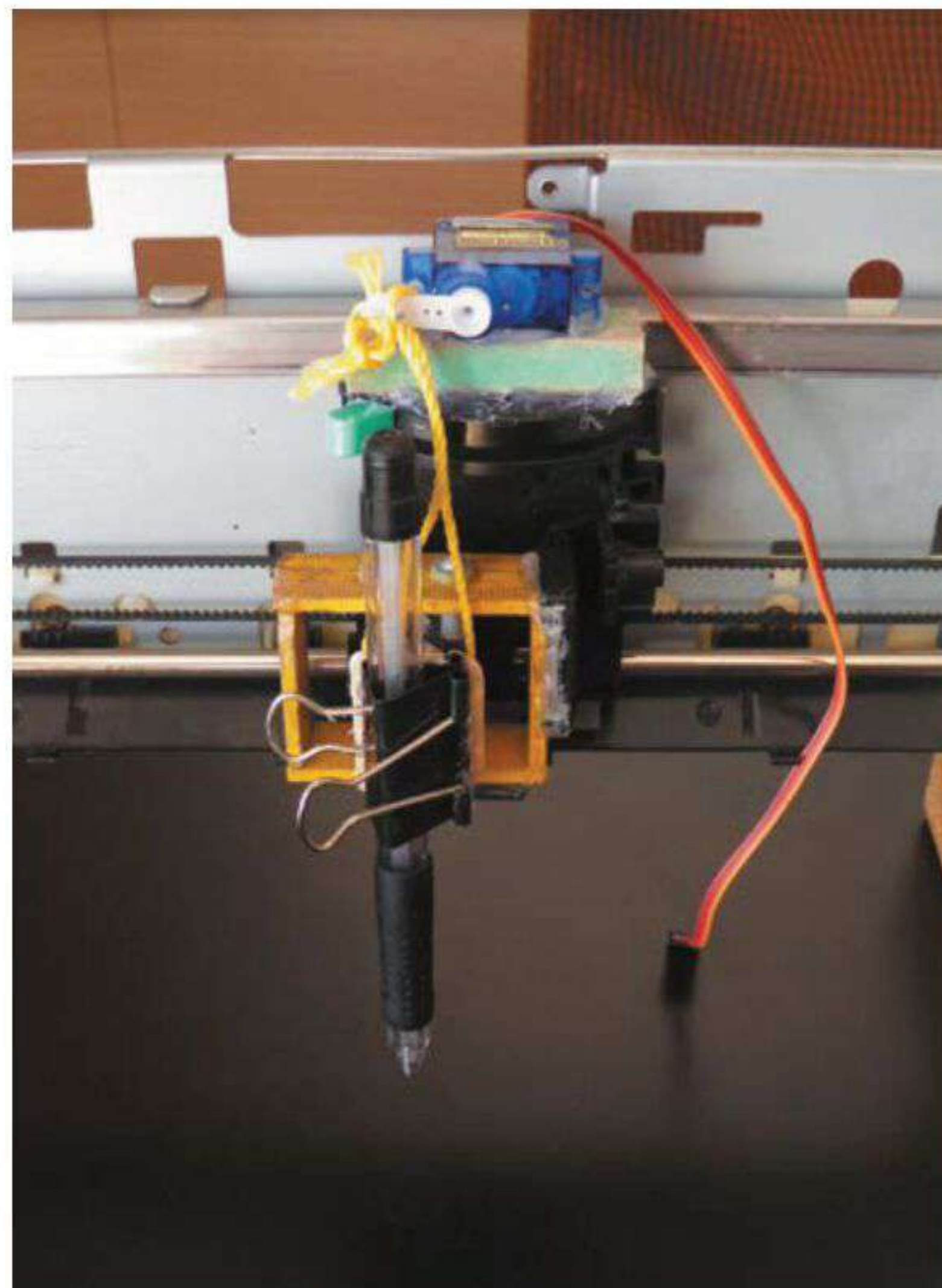
▶ The extra servo on top is to lift and lower the pen. With some string, of course

▲ Lifting up the drawing surface reveals the old scanner parts below, as well as the platform's stepper motor
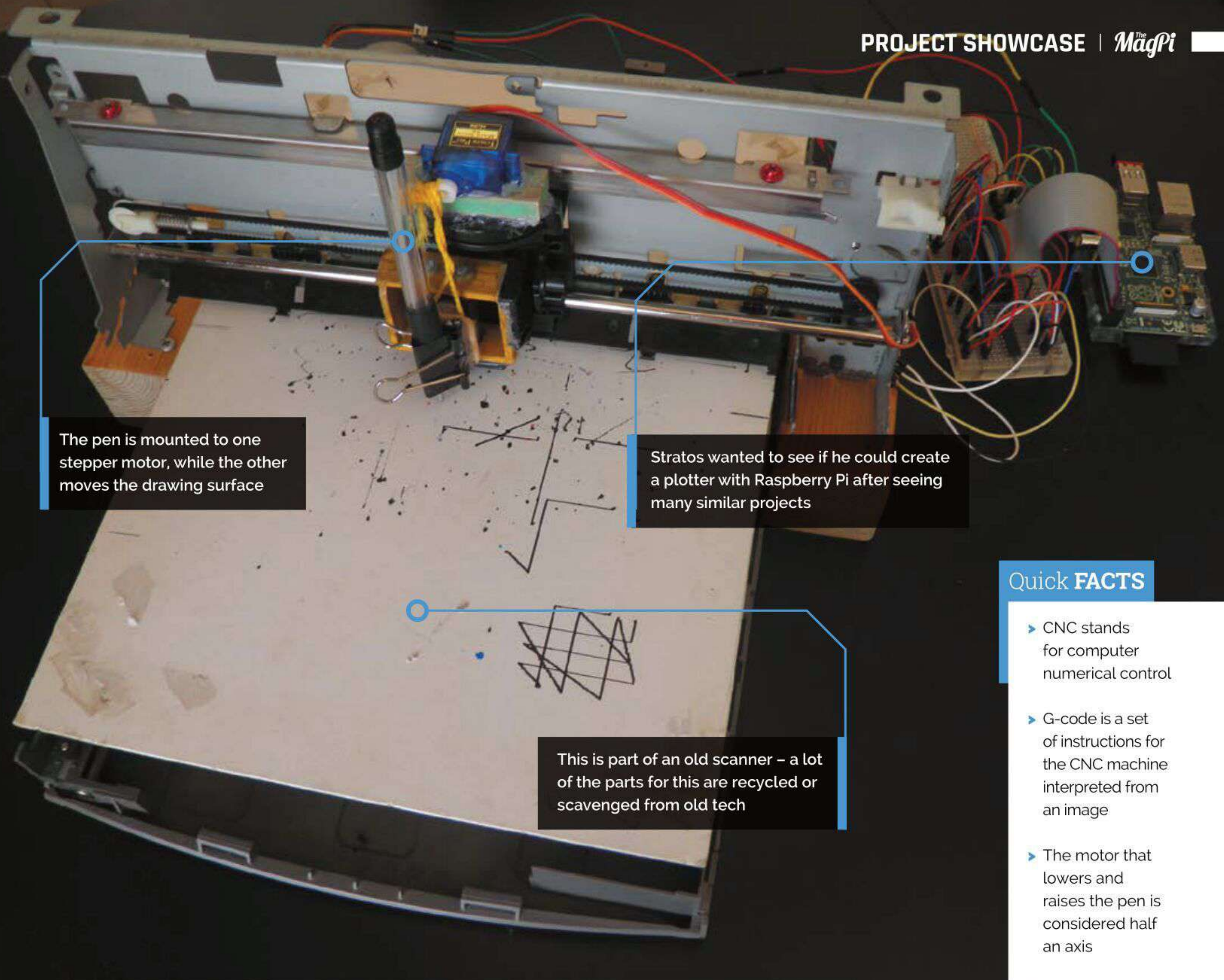
The pen is mounted to one stepper motor, while the other moves the drawing surface

Stratos wanted to see if he could create a plotter with Raspberry Pi after seeing many similar projects

This is part of an old scanner – a lot of the parts for this are recycled or scavenged from old tech

Quick FACTS

> CNC stands for computer numerical control

> G-code is a set of instructions for the CNC machine interpreted from an image

> The motor that lowers and raises the pen is considered half an axis

> A stepper motor is used, as its movement is performed in precise fractions of rotation

> Stratos usually works in Java

plotter to draw a diagonal line. I had been trying several algorithms in Python for a long time, but eventually the simplest one worked how I wanted."

Recycling to work

The end result is a little robot that can draw – exactly as planned. You can see it in action on YouTube here: **magpi.cc/cncplotdraw**.

We're big fans of recycling and upcycling for projects here, and recycling was always part of Stratos's plan. "I wanted to minimise the cost of the project as much as possible in order to find out how cheap it can be," he says. "That is the reason that I reused parts from an old scanner and a printer for the hardware part. Also, I used L293D chips instead of the [more expensive] L298D motor driver board, so the only cost was actually a Raspberry Pi and its accessories, which I owned anyway.

"Moreover, I implemented the software program myself because I wanted to find out the internal working of a CNC plotter. So I would say the only thing that it cost me mainly was my time, which I enjoyed spending while doing this project!" M

> " I had seen people on the internet creating and using CNC plotters and always wondered how these machines work "
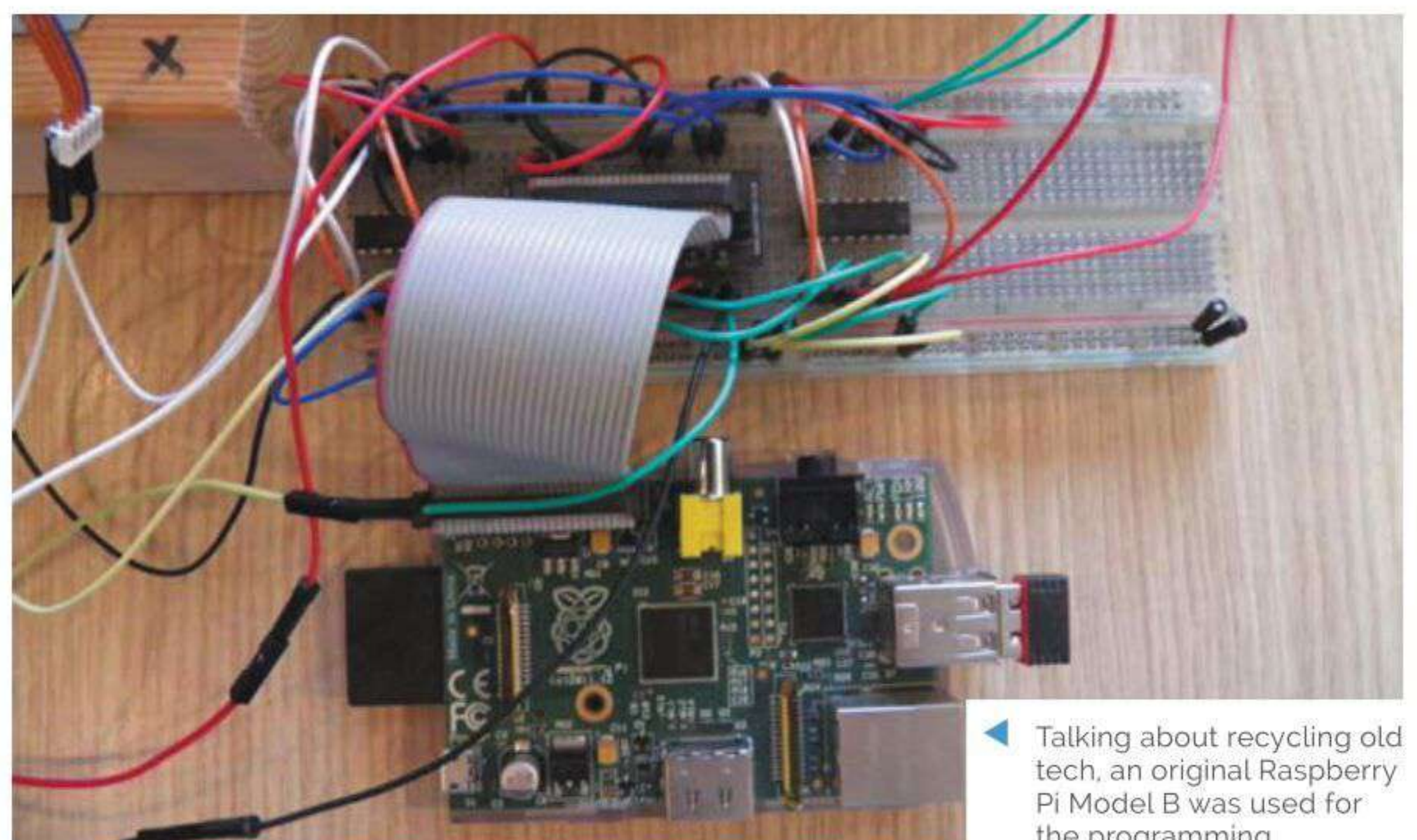
Talking about recycling old tech, an original Raspberry Pi Model B was used for the programming

RFID Gro Clock

Live-wire toddlers are exhausting, with no concept of bedtime. A Raspberry Pi-based 'Gro Clock' signals when it's time to sleep. Sounds like bliss to **Rosie Hattersley**

MAKER

David Gardner

David is an enterprise IT technician. He recently repaired several household electrical appliances, including a 1970s food mixer, a Kindle, and his Sonos audio streaming system (to which he eventually hopes to add more features).

magpi.cc/
rfidgrowclockgit

Getting enough sleep when you've got small children can be a challenge. David Gardner took a practical approach to sleep deprivation, devising a clock that uses a traffic light system to let the kids know when it's OK to get up. The RFID Gro Clock is based around Raspberry Pi Zero W and has a custom-made 3D case. The project took about six weeks to complete, and was finished just in time for Christmas.

Man with a plan

The aim of David's RFID Gro Clock project was "to get my son to be more independent in going to bed and then also to stay in bed longer in the morning. From a purely selfish point of view, that would give me a bit more time in bed," he notes.

To entice his three-year-old to go to bed in the first place, he decided to provide "some form of entertainment." He also needed "a method to show somebody who cannot tell the time when it is OK to get up."

Story books that mentioned CDs piqued his son's interest, so David decided this was a good option for the entertainment element. Using RFID as the control mechanism (for MP3s and other audio files) also made using the Gro Clock more intuitive: "I don't like my children having lots of interactions with screens, so this is a great, physical way for kids to be able to control things."

Building blocks

David based the project around Raspberry Pi Zero W for its GPIO programmability, memory, and microSD card support, as well as its compact size and low cost. He used Python to code everything and decided to use VLC Player for the MP3 playback, "as this has a pretty well-documented API and Python library, plus support for playing audio CDs."

He added a 'setcd' command to identify the number of tracks on a CD when it was inserted, and used events in his Python code to understand when the next or previous track was being played.
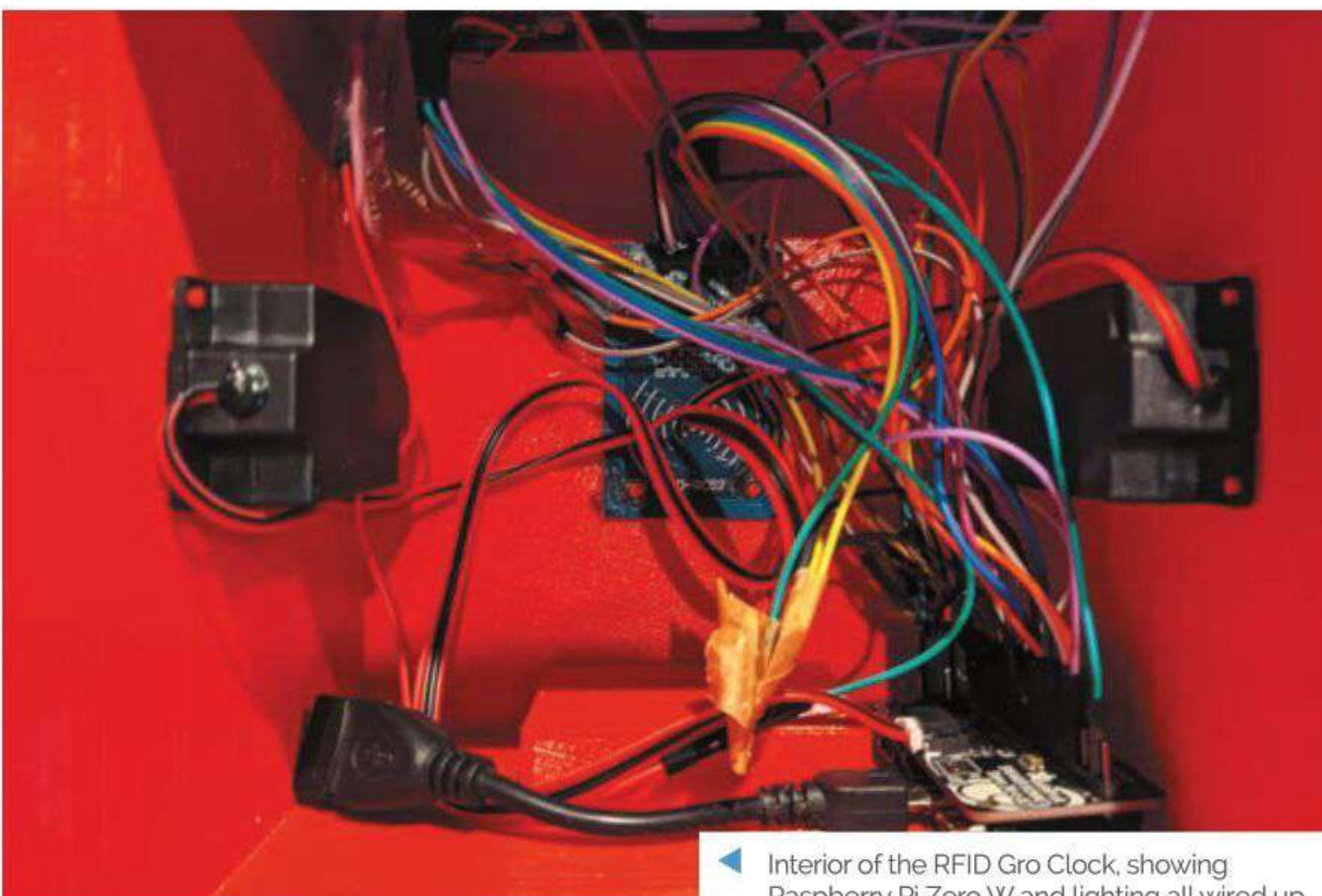
◄ Interior of the RFID Gro Clock, showing Raspberry Pi Zero W and lighting all wired up

► David used FreeCAD to design the 3D-printed case for his Gro Clock

On-screen icons provide further indications of whether it's playtime or time for a nap. The clock will be handy once their toddler learns to tell the time

A traffic light system indicates whether it's time to sleep, to read, or listen to music in bed, or OK to get up and wake the parents. Sneakily, the parents can adjust the times and arrange a lie-in

A CD player for audiobooks and music extends the Gro Clock's usefulness. The large square buttons play, skip, pause, and replay tracks

❝ This is a great, physical way for kids to be able to control things ❞

He advises anyone planning a similar project to do their research and planning first. For example, he has no 3D printer, so used a 3D printing website. Having created a design in FreeCAD (using YouTube videos as a guide) and sent the resulting STL file to print, David realised he'd omitted two (thankfully non-critical) items – a potentially pricey mistake since 3D printing was already under way.

"Raspberry Pi has been great for the project. It allowed me to have the flexibility of a computer and all the software packages that are available, whilst also giving me the ability to interface with a wide variety of electronics components," he reveals.

Because he was using existing components as far as possible, not everything came together immediately. The RFID element caused a few issues with power consumption due to the Python package chosen, and because it and the OLED connect via SPI. Nonetheless, David recommends RFID cards as a method of control. "It's a great way to allow little people to interface with devices," he says. "Maybe when [my son] gets a bit older and more into music, I may look to have some RFID cards play Spotify songs."

▼ Testing the components before installing them in the custom-designed case
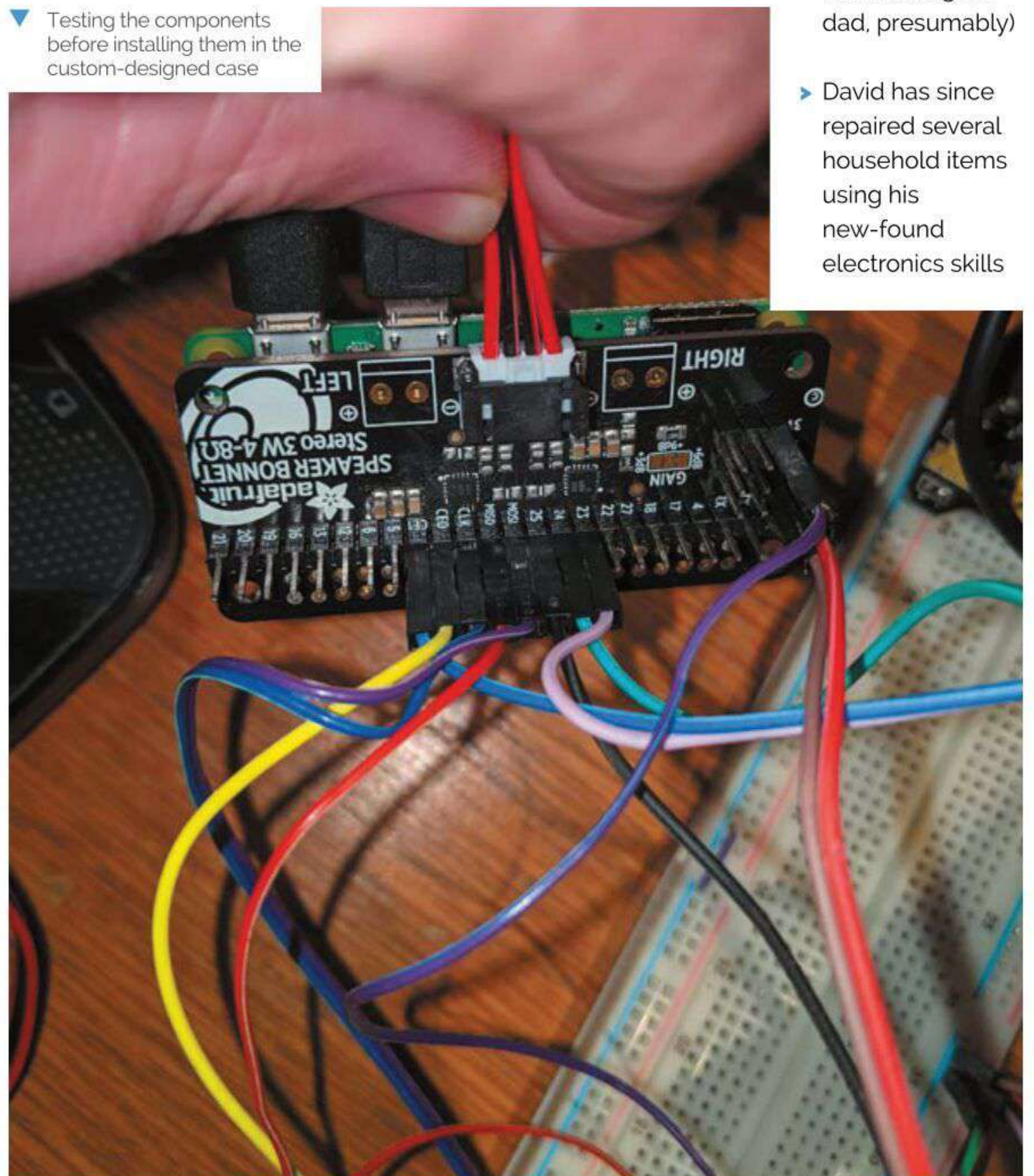
Real-time bee monitor

Researchers are buzzing with excitement after devising a low-cost method of monitoring wild bees, as **David Crookes** discovers

MAKER

Michael Smith

Michael is a computer scientist at the University of Sheffield and the lead author of a study into the monitoring of bee species in the UK, using relatively inexpensive technology.

magpi.cc/ beetrack

Scientists seeking to better understand the ecology of bees know that monitoring them in the wild is no easy task. Harmonic radar – the best way of tracking bees – is expensive and complicated, which is why researchers are currently abuzz over a new method that puts a Raspberry Pi computer at its heart.

Led by computer scientist Michael Smith, a team of researchers from the University of Sheffield and The Bumblebee Conservation Trust have figured a way to make the striped insects easier to spot. They're dressing bees in hi-vis retroreflective vests and taking photographs of the environment, before subjecting them to a machine learning model that operates in real-time.

"I was reading books by Dave Goulson, who described the problem of finding the nests of bees, and it got me thinking of ways to spot them from a distance without needing an electronic tag," Michael tells us.

"When I was cycling home one evening," he continues, " I noticed how retroreflectors are very noticeable when lit by the blinking bike light. It was a eureka moment."

Bee-hold Raspberry Pi
Michael devised a method in which two photographs would be taken of an environment – one using a camera flash and the other without. He experimented by connecting a Raspberry Pi 3

> " Machine learning helps to remove false-positive spots caused by other objects "

to an industrial global electronic-shutter camera, but soon switched up to a Raspberry Pi 4. "The better CPU meant we could process images much faster and the extra memory improves the image analysis as more images can be processed at once," he says.

Study co-author Mike Livingstone is catching bees from the researchers' nest in order to tag them

The method depends on being able to take a flash photograph, so the camera needs to be able to expose the entire sensor at once, not just scan lines. "The very short exposure you can get with the electronic shutter (down to one microsecond) means I can match the exposure to the length of the flash, which is a few microseconds," Michael reveals. "It means almost all of the illumination in the photo is from the flash, even on a bright sunny day, and so it's easier to detect the retroreflector."

Hive of activity
The machine learning process subtracts one photo from the other, leaving an image containing bright spots if the retroreflector-wearing bees happened to be in the frame.

"Machine learning helps to remove false-positive spots caused by other objects such as moving trees and litter," says Michael, who collected the machine learning data with two of his students – Isaac Hill and Chunyu Deng – by walking around in front of the tracking system with a reflector on the end of a stick.

▲ Since the tracker works best when it's looking down and is away from clutter, the scientists have experimented with ways of getting the system in the air, trying a hexacopter UAV, a 10 metre mast, and a tethered balloon

Two to four camera flashes (TT560 models by Neewer) are used. A Veroboard with mounted transistors allows the flashes to be triggered from the GPIO pins

Real-time data is provided via a webpage (and flask API) which is hosted on a Raspberry Pi 4

The £400 GCC2062M camera takes images which are then overlaid with the bee's location, highlighted by a cross

Quick **FACTS**

> Retroreflective tags are attached to bees

> The tracker spots bees up to 40 metres away

> It monitors foraging behaviour and 3D flight paths

> Seven species of wild bee have been monitored

> Each tracker costs around £500

▼ The retroreflective tags placed on the bees are made of the same fabric as the high-visibility vests worn by cyclists

"To build the system, we manually labelled where our reflector was in the photos afterwards. These labels, combined with false-positive dots in the same images, were used to train the classifier, and we used Raspberry Pi OS, Python 3, standard libraries, and the Aravis library to interface with the camera and process the results."

So far, the team has been able to detect bees from up to 40 metres away and this has thrown up some surprising results. On one occasion they found buff-tailed bumblebees up a pine tree some 33 metres distant in a location the researchers wouldn't have usually looked.

"We've used the trackers in gardens, fields, and at various places on the university campus, but we're in touch with other researchers who will be using them for looking at the initial flight of bees as they leave nests, or for monitoring bees foraging inside glass-houses. It also makes sense to think about tracking and detecting other insects. There are a lot of open research questions in behavioural entomology." Ⓜ

JOIN FOR 12 MONTHS AND GET A

FREE Raspberry Pi Zero W Starter Kit

WITH YOUR FIRST 12-MONTH SUBSCRIPTION

Subscribe in print today and you'll receive all this:

WORTH £20

▶ Raspberry Pi Zero W

▶ Raspberry Pi Zero W case with three covers

▶ USB and HDMI converter cables

▶ Camera Module connector

This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.
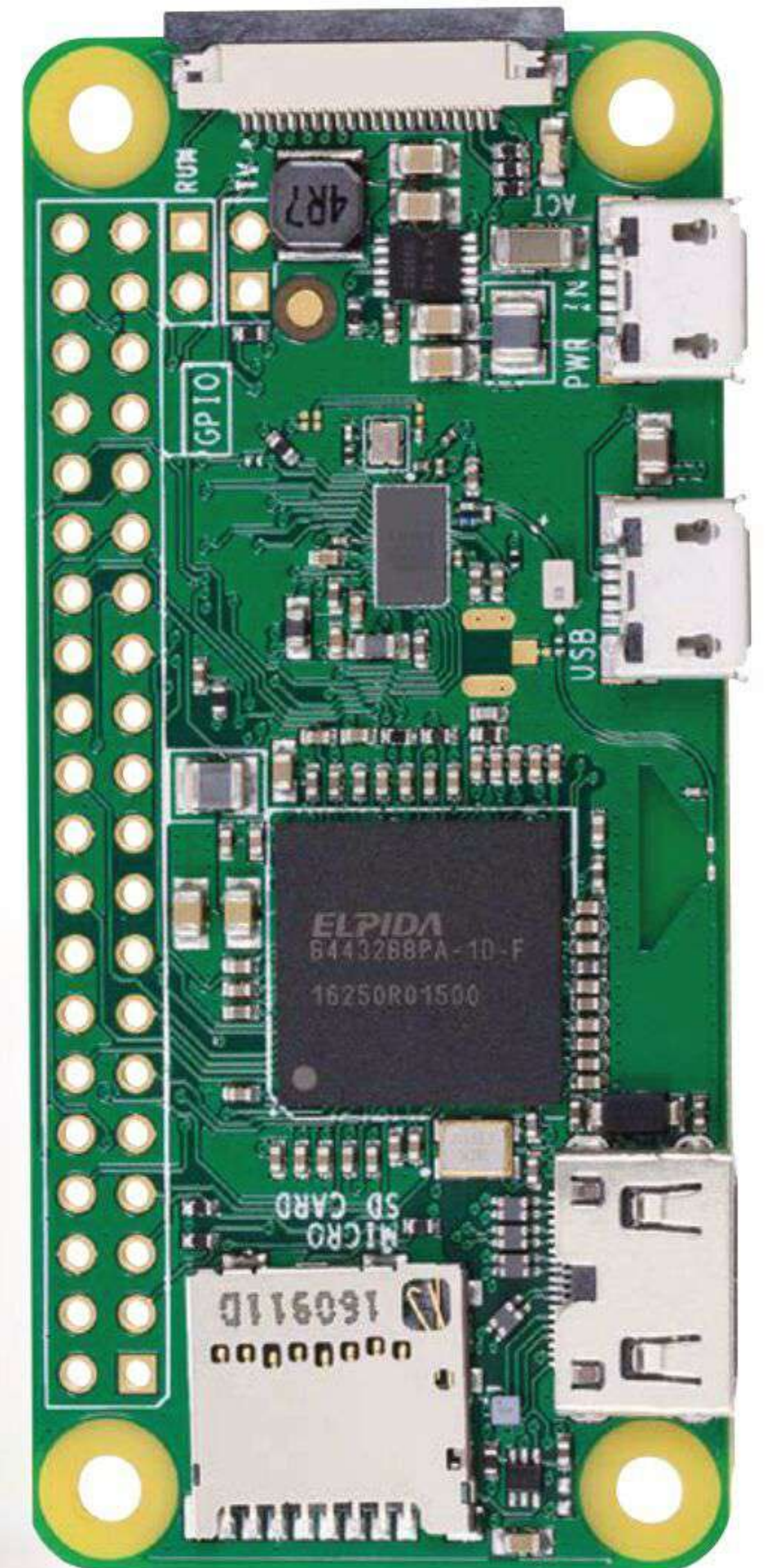
⊙ Buy now: magpi.cc/subscribe

SUBSCRIBE on app stores

From **£2.29**

Available on the App Store

GET IT ON Google Play

#MonthOf MAKING
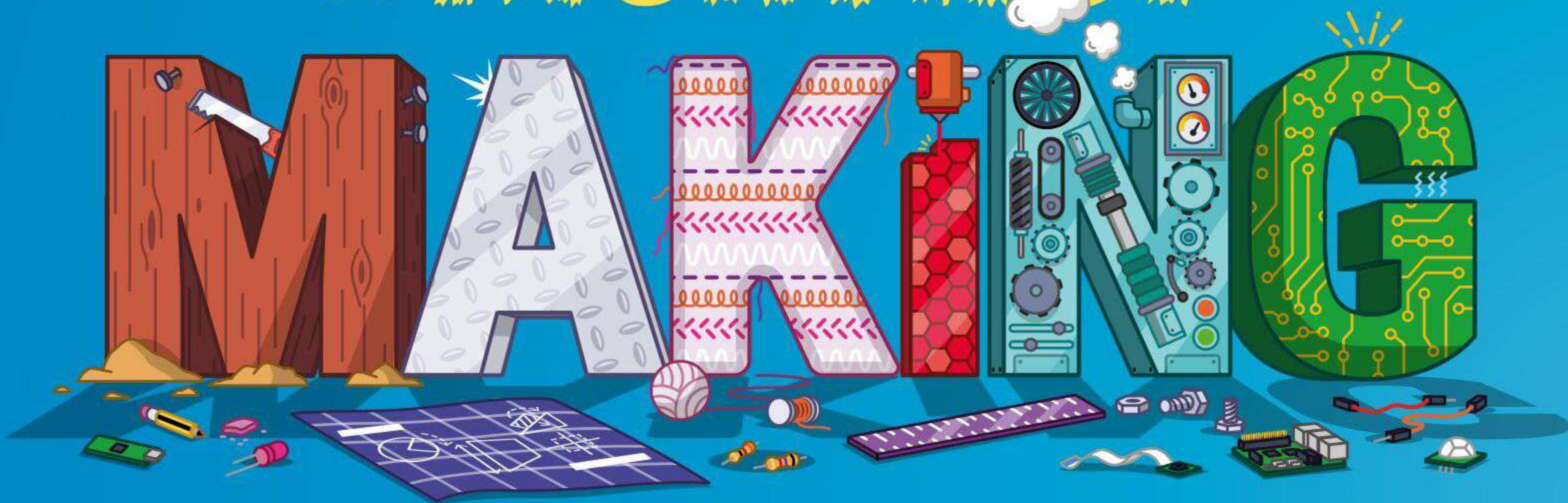
Try out other ways of making with **#MonthOfMaking** 2021 and The MagPi!

It's that time of year again! March around these parts is the #MonthOfMaking, our month-long community building event.

We love to seek out and showcase the amazing projects the wider Raspberry Pi and maker communities come up with – and by wider, we really mean wider. In the mag we often talk about electronic-based projects – we are a computer magazine, after all. However, the world of making encompasses much more than that: crafting, sewing, knitting, baking, sculpting, photography, model-making, cosplaying, and even more.

While you can make anything you want for #MonthOfMaking, we want to highlight other ways you can make and be creative in this issue. Let's build!
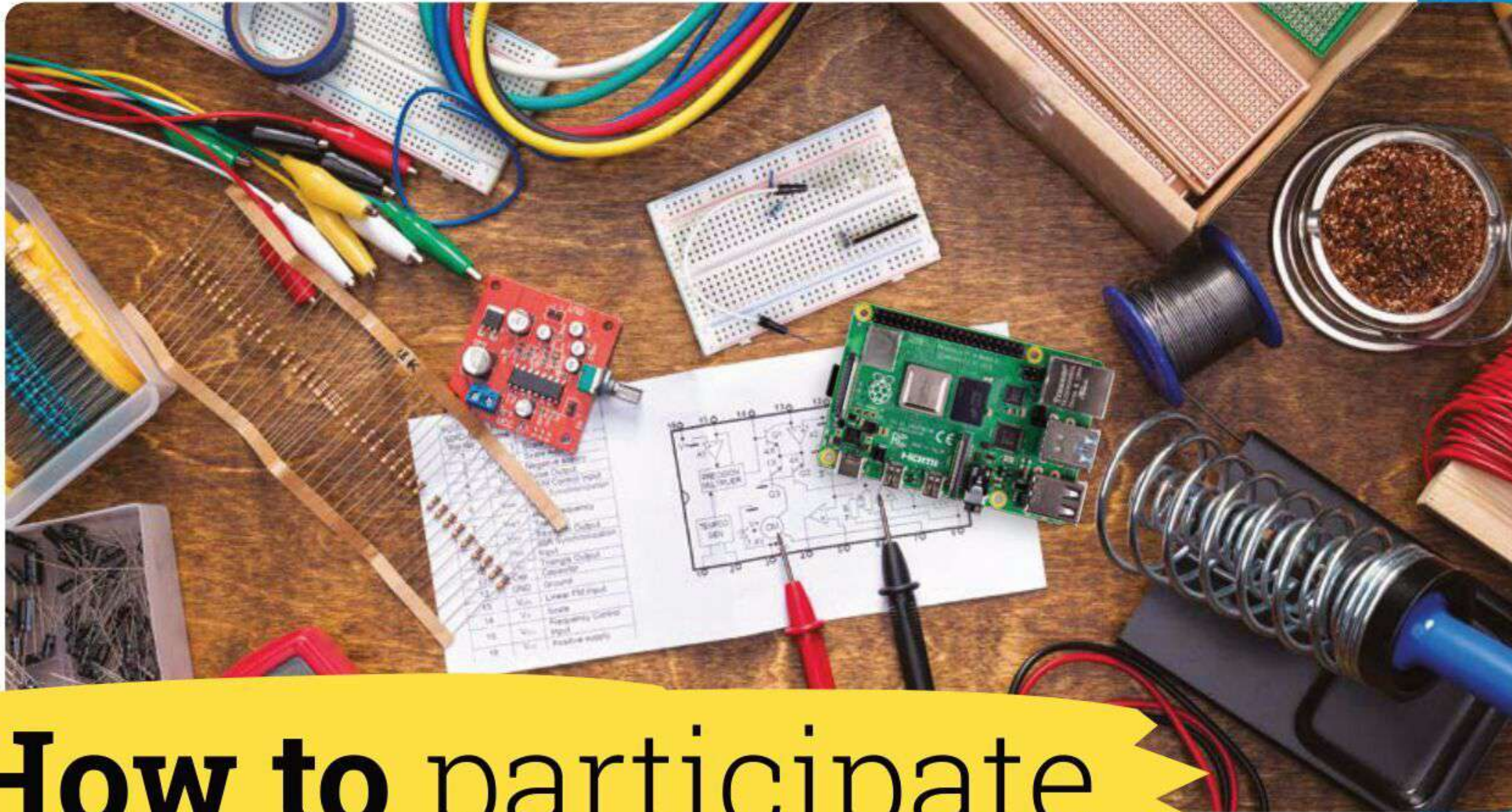
How to participate

The rules of #MonthOfMaking are simple

1

Work on a project, new or old

2

Take photos of your progress and completed projects

3

Share it to Twitter and/or Instagram with a helpful description

4

Make sure to include the #MonthOfMaking hashtag

#MONTHOFMAKING RESOURCES

The MagPi 79

We introduced #MonthOfMaking in *The MagPi* #79 with eleven projects you could try out

magpi.cc/79

The MagPi 91

In *The MagPi* #91, we showed you how to document and share your projects with the community

magpi.cc/91

HackSpace Magazine

Our sister publication, HackSpace Magazine, is joining in on the #MonthOfMaking fun this year. Its back catalogue is a treasure trove of amazing project ideas and inspiration

hsmag.cc

#MonthOfMaking hashtag

See what others are doing! Be inspired by each other, and offer help if you can when asked

Craft making

Sewing, knitting, crochet, weaving, etc. are some of the oldest forms of making. Here's how to supercharge them with tech

Usually the closest we get to doing some kind of craft project with Raspberry Pi is with wearables. While they are cool in their own way, and can be definitely be part of this subject, we don't do much that truly combines yarn craft and electronics. Here are some ways you can.

MAGPI MEDIA

We recently did a big feature on Arts and Crafts with Raspberry Pi, and you can find it in issue 101 of *The MagPi* (**magpi.cc/101**) for more tips and inspiration.
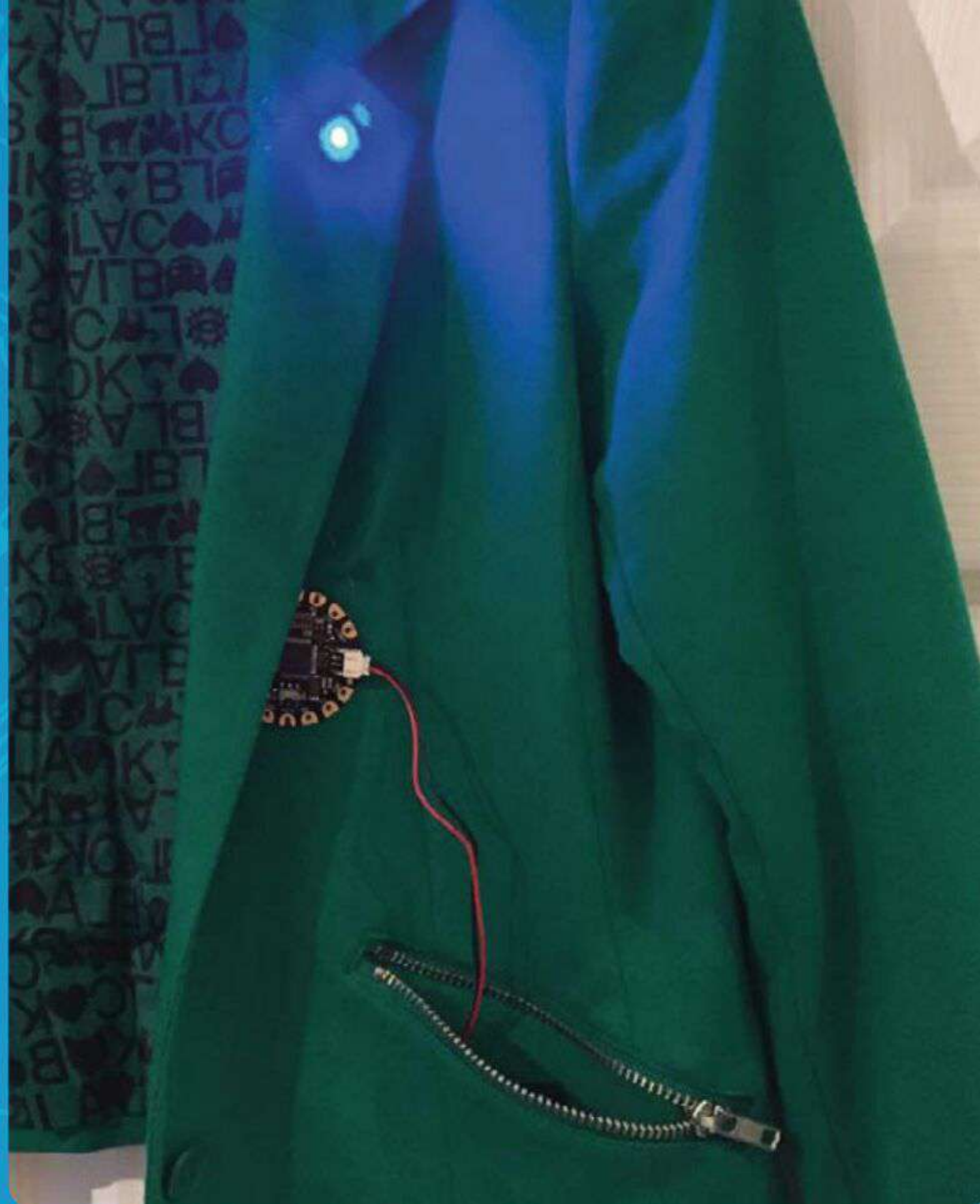
Getting started
with wearables

This is a great place to start if you want to learn a bit more about sewing and coding with conductive thread. It makes use of Adafruit FLORA; however if you want to take a stab at modifying it for a Raspberry Pi Pico, it's more than doable.

Sewing is something that does take a bit of practice but it's a really handy skill if you want to modify or even make your own clothes! Adding wearables in this fashion can be a lot of fun, and is also very useful when it comes to cosplay.

magpi.cc/wearables

TAKE IT FURTHER...

Social Media without the Internet

This project is more of an art piece, but it shows the extreme application of wearable technology with buttons and screens all over these coveralls. A high five counts as a 'like' in this offline social experiment. However, you can use some of the ideas for different types of costumes – even a Halloween get-up many months in advance.

magpi.cc/socialwear

CNC Embroidery machine

This is a cool mixture of tech. A lot of modern embroidery machines let you load up a pattern via a computer so that the end result is much more accurate and faster if you require it. This project is all about hacking a normal sewing machine that connects to a Raspberry Pi and some motors to do the hard work. It's also a lot cheaper, if you have the skills, materials, and patience.

This is a mix of hardware and software hacking and quite advanced. If you fancy doing some other kinds of upcycling/recycling, check over the page for other projects.

magpi.cc/cncembroid

ALTERNATE EMBROIDERY TIPS...

Conductive thread embroidery

Not sure if you want to hand-sew conductive thread into some fabric? Want to try to hide it? This ingenious method of attaching conductive thread by wrapping it in thread also allows the wire to be somewhat insulated! With some practice and a steady hand, you could possibly do this with a normal sewing machine as well!

magpi.cc/embroidery

WEARABLE TECH PROJECTS

Our friends over at HackSpace Magazine worked with the amazing Sophy Wong to create a fantastic wearables book, full of incredible ideas for merging tech and fashion. Check it out at **magpi.cc/ wearableprojects**.

Soft circuit button
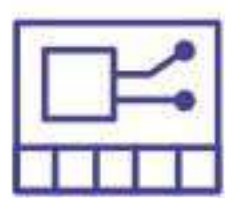
This project is a little more of a hacky/DIY way to work with fabric and electronics. Create a soft, safer button that can be used in a variety of wearable or other applications. It's also fairly disguised for a button.

There are also some good tips on how to use conductive thread in a sewing machine, such as lowering the tension on the machine. This project is based on Arduino, but it's a perfect fit for Raspberry Pi Pico.

magpi.cc/softbutton
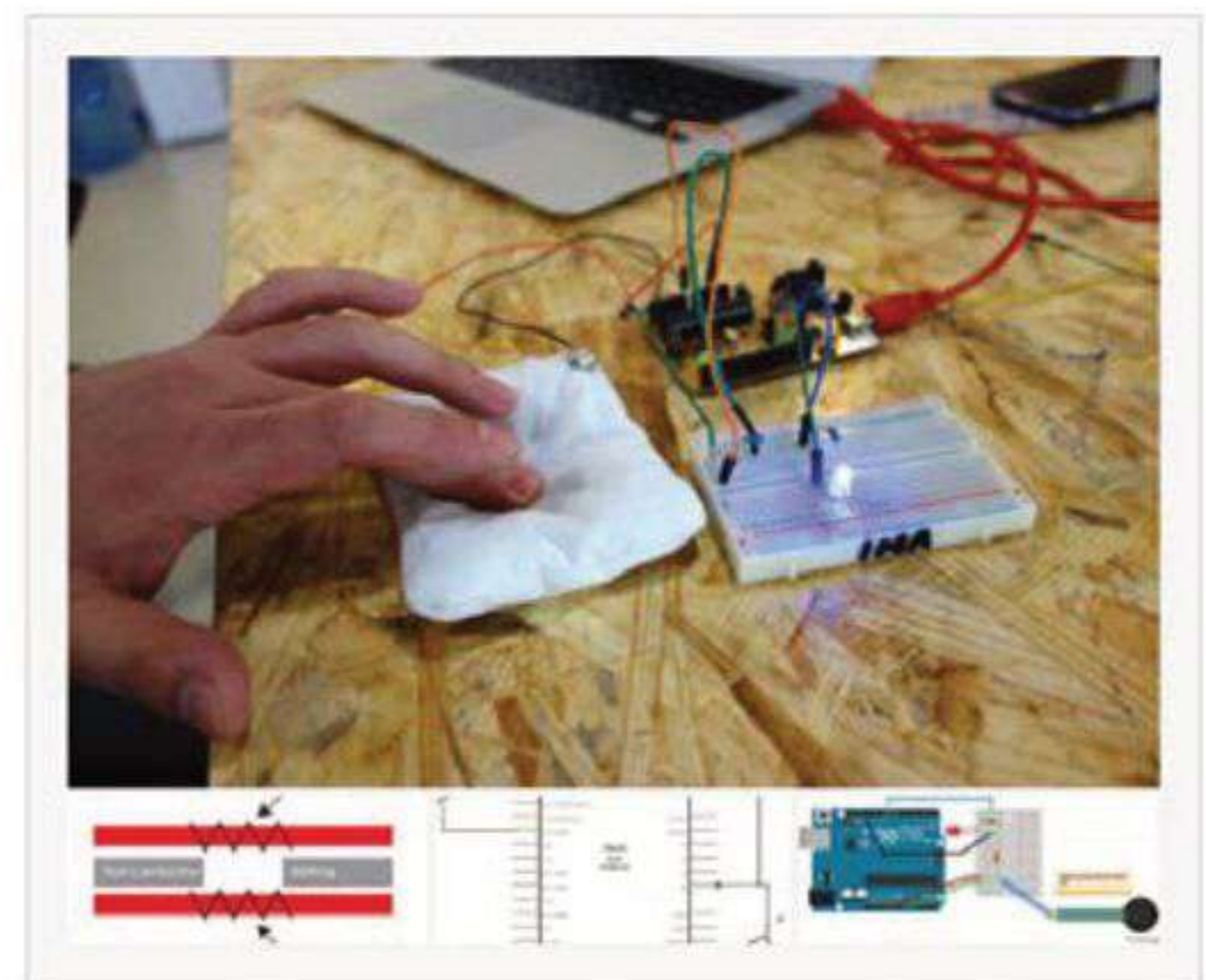
Recycling
& upcycling builds

Use old tech and unwanted household items and give them a new life

Making can get expensive, but there's so much you can do with old tech or a cardboard box with a little ingenuity and elbow grease. Here are some great projects that might get you inspired to turn your Amazon box into a robot lawnmower.

We regularly love to talk to Martin Mander about his upcycling projects, so we decided to pick his brains for some excellent upcycling tips. Check it out in issue 70 of *The MagPi*: **magpi.cc/70**.

Recycle

NeoPixel Recycled
Chandelier

Made from recycled bottles for an ultimate DIY look, this NeoPixel-powered chandelier is wonderful. It even uses recycled wood for part of the build. The mixture of bottles is not perhaps quite our preference, but it looks good either way. This project makes use of Adafruit boards rather than Raspberry Pi, but it's easy to swap them out so you can continue using what you have instead of buying anything new.

magpi.cc/chandelier

Recycled
Robotic Arm

A robot made from recycled parts and/ or scrap seems very much like a scrappy sci-fi movie character thing. However, this version is very real! It reminds us a lot of the MeArm robot kit and you could probably use parts of that to make this bigger robot.

Once again, this is all powered by Arduino, but controlling motors and servos with Raspberry Pi is incredibly easy using various motor or robotic HATs. Pairing a game controller will be a bit easier as well. The potato counterweight is universal, though.

magpi.cc/recyclearm

Upcycle

Game Boy Zero

Sticking a Raspberry Pi Zero inside a classic game controller or handheld – or any old tech, for that matter – is a perfect fit for the tiny computer. While there are many Game Boy-style cases and 3D-printed builds you can do (and they'll work very nicely as well), this version uses an actual original Game Boy, albeit with some external modifications, like extra buttons taken from an NES controller.

A cartridge has also been made into a giant microSD card adapter, so you can switch out cards easily.

magpi.cc/gbzero

Coffee Maker Greenhouse

Is this an upcycle or a recycle? It's probably closer to recycling as it doesn't really work as originally intended, but it does use old tech in a new way. Either way, it's very cool, as it's been turned into an automated planter waterer using the original functions of the machine.

The system can sense when the soil is dry, although currently it just waters on a timer. The brew button on the original chassis can be used to dispense water ahead of schedule as well, which is something we'd like to see more of in automated projects.

magpi.cc/greencoffee
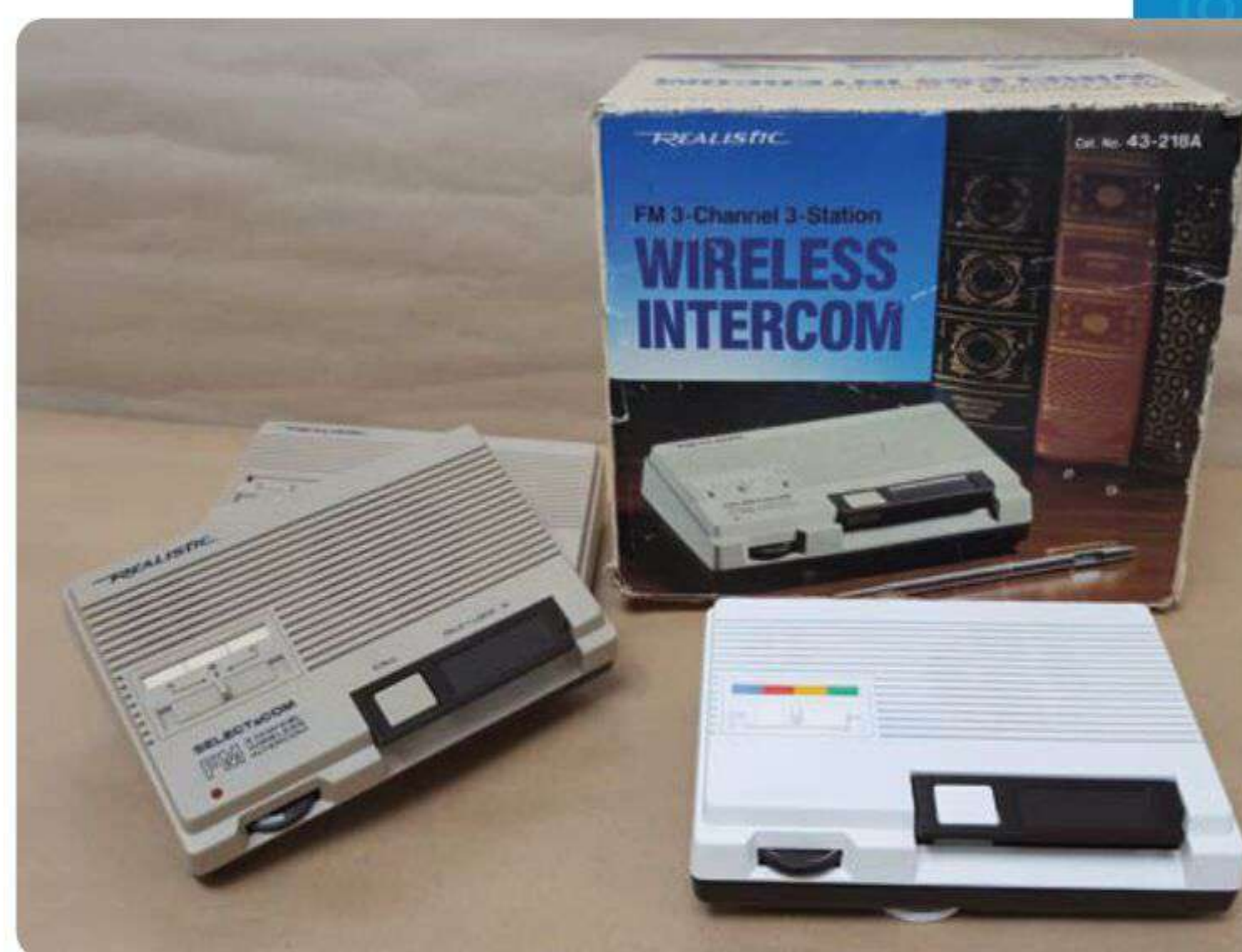
Google Pi Intercom

It would be very easy to fill pages with upcycling projects from the always-inventive and creative Martin Mander. This Google Pi Intercom remains one of our favourites due to its mixture of old tech (an eighties office intercom) and new tech (Google voice assistant software). He made it using the Google AIY Voice Kit given away for free with issue 57 of *The MagPi*, but it doesn't require that to make it work these days.

You can find second-hand and broken tech on pretty much all good online auction/selling sites – just be sure you've fully discharged any capacitors before playing with it.
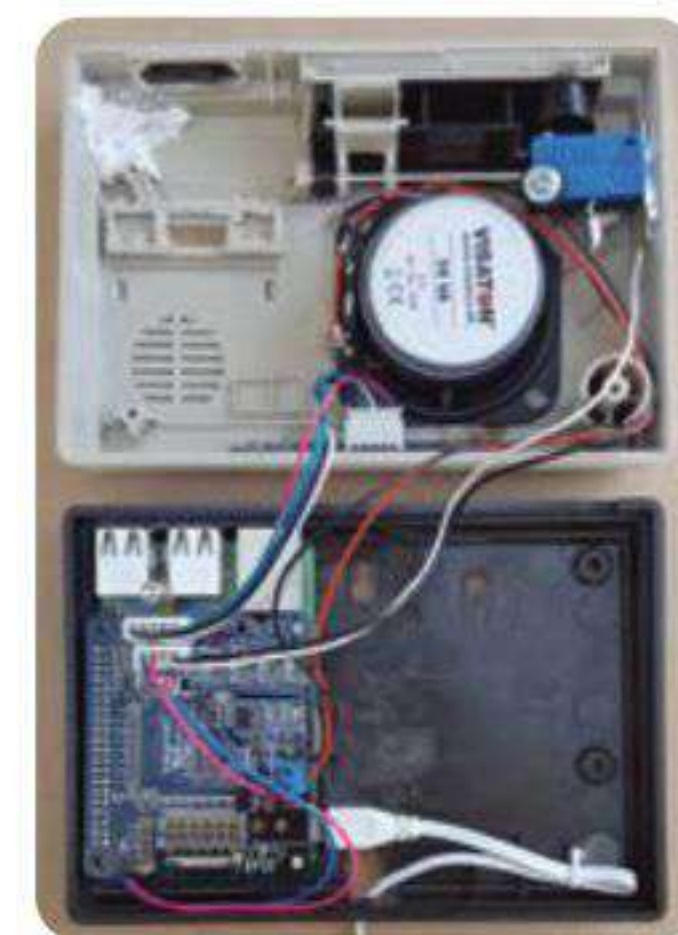
magpi.cc/piintercom

Raspberry Pi Photography

Take photos with Raspberry Pi, and even do animation with it!

Photography is an art and a science. Tweaking the settings of a camera for that perfect composition goes hand in hand with creating excellent photos. You can do so much with a Raspberry Pi and a camera – whether you use an official Camera Module / High Quality Camera, a USB webcam, or a hacked DSLR.

MAGPI MEDIA

Learn everything you need to know about using official Raspberry Pi camera boards with *The Official Raspberry Pi Camera Guide*: **magpi.cc/cameraguide**.
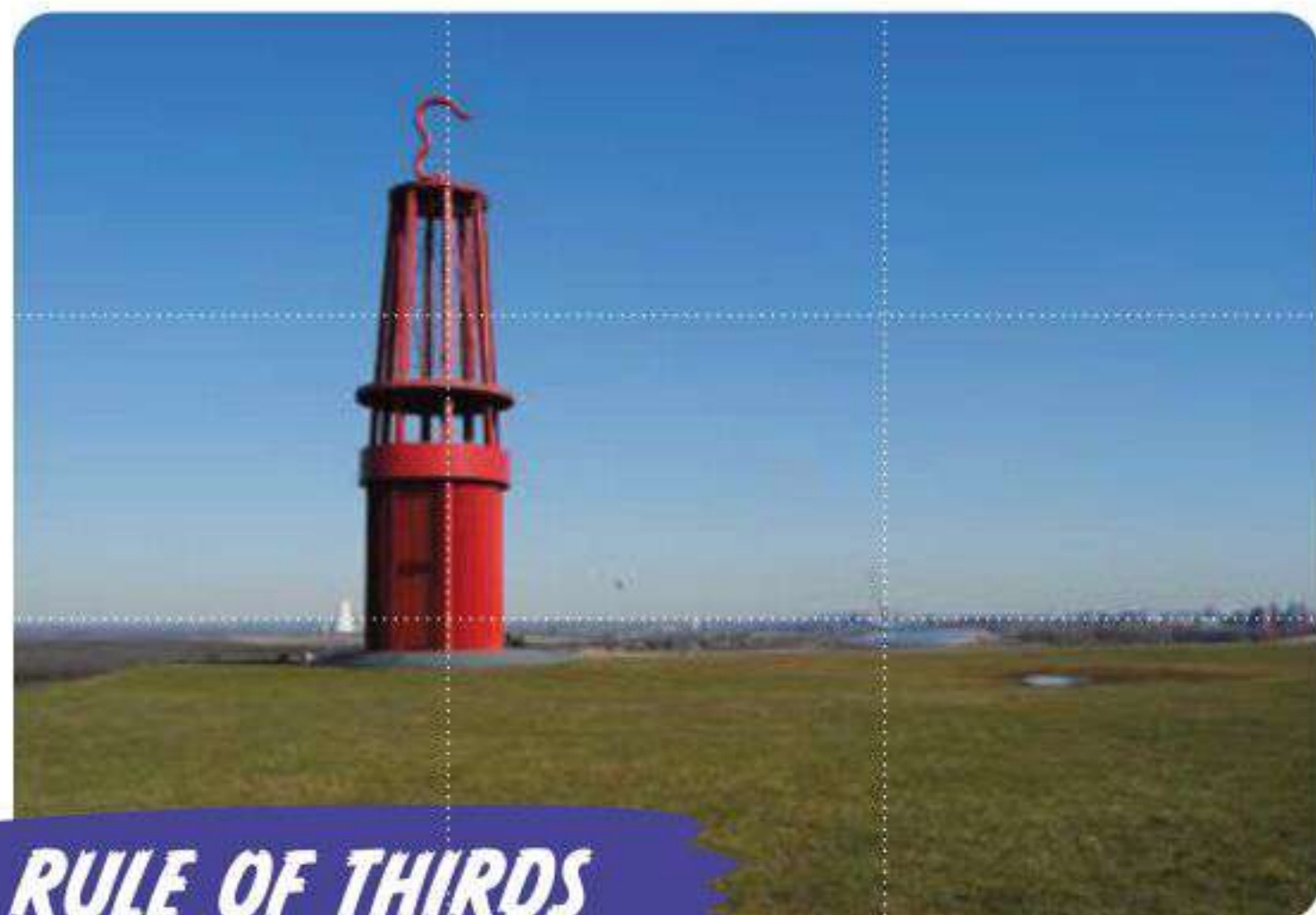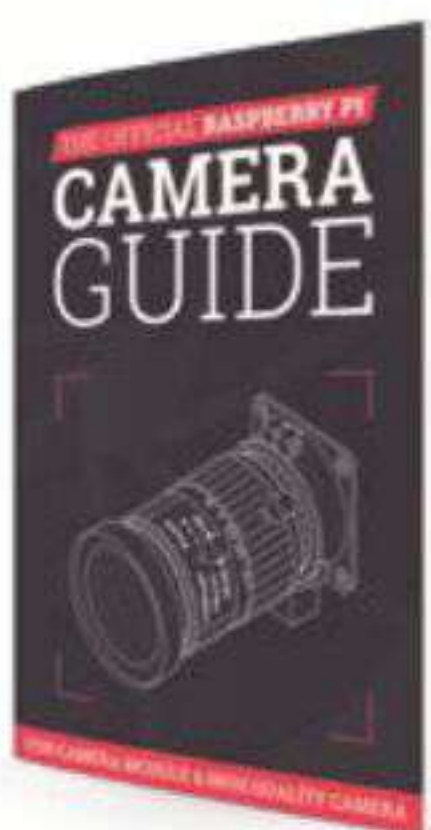
RULE OF THIRDS

A quick and basic way to frame your photo is to think of it in thirds – as in an equal 3×3 grid bisecting your photo, like the image above. Features of your photo, such as the horizon or a tree, should lie roughly along one of the lines or an intersection of the lines to make them stand out more.

High Quality
Camera case hack

Want an easier way to transport a Raspberry Pi HQ Camera and use it as a more traditional point-and-shoot camera? The Raspberry Pi Foundation's Richard Hayler and family show us with this excellent hack using an official Raspberry Pi case.

It's a permanent hack using some drilling, so make sure you don't mind cutting into a spare case – and have adult supervision around if you need it!

magpi.cc/94

Make a 3D camera

While you can do some fancy digital tweaks to a single photo to make it more 3D, the original and best way is to just take photos at the same time, using lenses that are side-by-side. Regular contributor to *The MagPi* PJ Evans shows you how using two High Quality Cameras and two Raspberry Pi Zero boards linked together.

He even demonstrates how to view the photos – something that will come in very handy for impressing friends.

magpi.cc/3dcamera
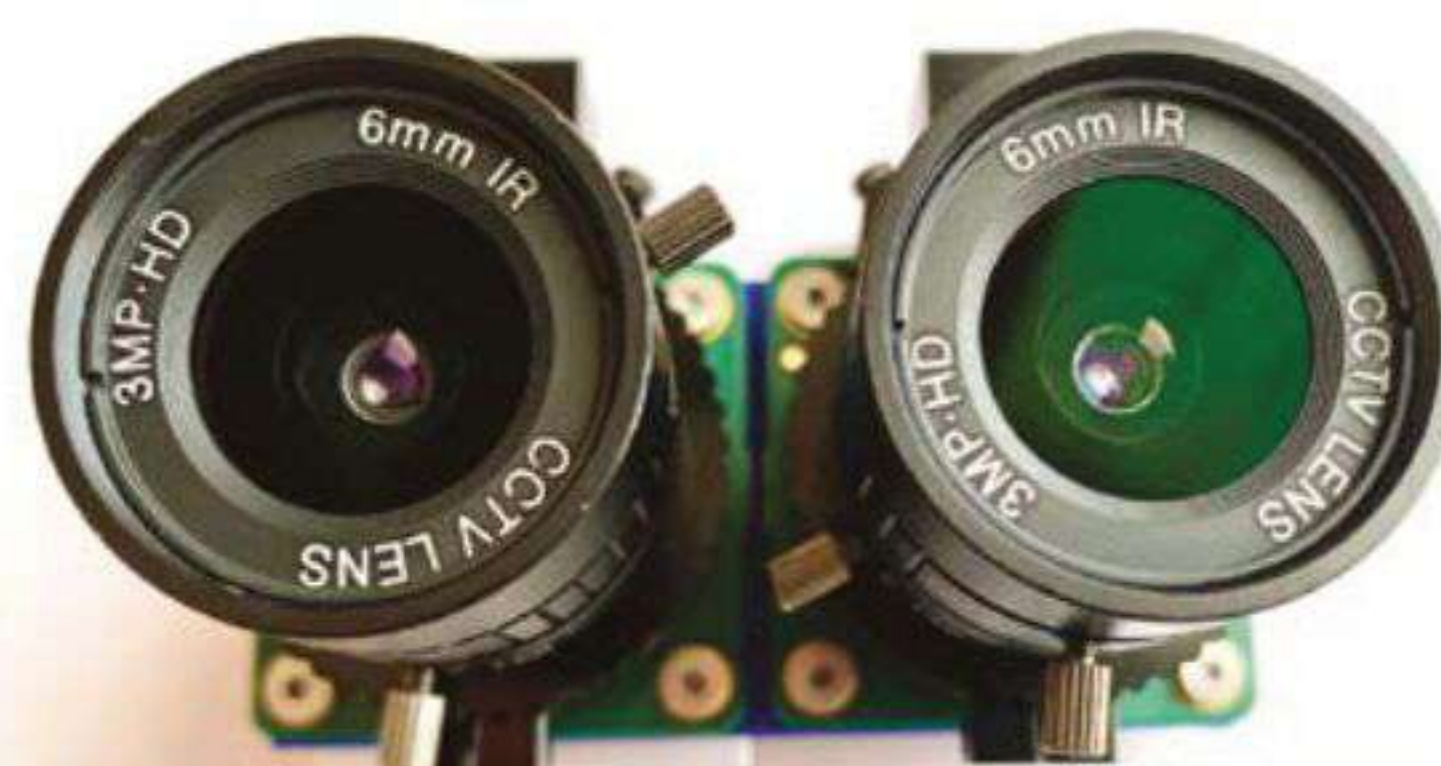
FRAMES PER SECOND

When creating an animation, ideally you want to aim for 24 frames per second, which is the traditional frame rate of cinema. This is called 'animating on ones'. That can be a lot of work, though, so some animators will skip a frame or three if it will save time.

Frames per second	New picture every	Name
24	Frame	Ones
12	Second frame	Twos
6	Fourth frame	Fours

There are lots of other animation tricks, like moving backgrounds in parallax to get a sense of motion, or shining a light behind a picture with cut-outs to create an illusion of sparkles.

Simple Raspberry
Pi Camera Trap

This project does a bit of recycling, using a plastic food container as an airtight and weatherproof box, so also possibly belongs in the previous section of this feature. It's a pretty simple build, technically, but it does make great use of code to get it to work just right.

It works by waiting for motion before taking photos or movies, thanks to MotionEyeOS, software mainly used for CCTV. The principles are very similar, though.

magpi.cc/cameratrap

Stop-motion
animation

This project from the Raspberry Pi Projects site is very simple yet very effective, allowing you to take continuous shots using the press of a button before combining them all into one video at the end. There's also a similar project in the *Camera Guide*. Creating sets and characters using the craft and recycling sections might be a great idea as well.

magpi.cc/stopmotion

Motion
time-lapse camera

There's a time-lapse photography project already in *The Official Raspberry Pi Camera Guide.* This particular setup takes it a step further, however, allowing for precise movement as part of the time lapse.

It uses a mix of Arduino and Raspberry Pi, along with some clever code, to control a camera and motor to get a smooth pan-and-tilt effect on a time-lapse video. It should be easy to upgrade with the HQ Camera for more stunning shots.

magpi.cc/motionlapse

Get started with

MicroPython
on Raspberry Pi Pico
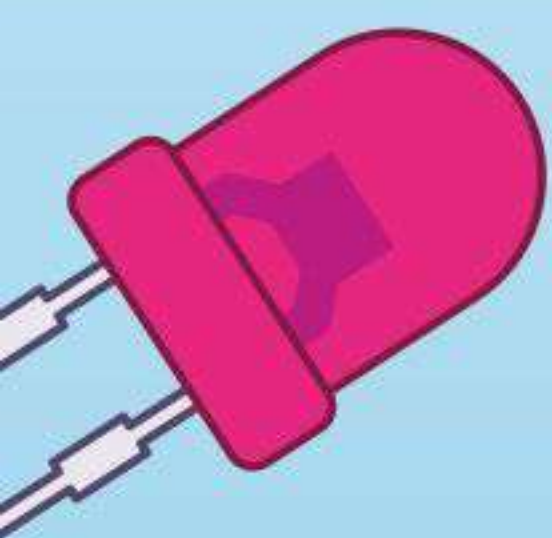
by Gareth Halfacree
and Ben Everard

Build a Pico
Pomodoro timer

Stay focused and get more done with help from
Raspberry Pi Pico and 112 colourful LEDs

MAKER

Nik Rawlinson

Esperanto-speaking, pencil-wielding, single-board computing fan who likes hyphens and remembers what that icon on the save button depicts.

nikrawlinson.com

The Pomodoro Technique – named after a tomato-shaped kitchen timer – has helped countless procrastinators get more done by breaking down jobs into 25-minute bursts, followed by five-minute rests. Although there are plenty of apps that will count down each work and rest cycle, picking up your phone to check how long remains can be a distraction in its own right.

But pairing a Raspberry Pi Pico with the bright LEDs of a Unicorn Pack means you can see at a glance when your next break is approaching and, if you keep it just out of your eyeline, that warm red glow is a reminder to carry on working.

You'll Need

> Pico Unicorn Pack
magpi.cc/ picounicornpack

> Pico Header Pack
magpi.cc/ picoheaderpack

> Soldering iron and solder
magpi.cc/ solderingiron

> Thonny IDE
magpi.cc/thonny

01 Give your Pico some pins

We want to attach Pico to a hardware array of LEDs – but, as Pico lacks the GPIO header of a regular Raspberry Pi, we'll first need to solder a header to the long row of holes on either side of Pico itself. With the USB socket uppermost, fit Pico over the shorter pins on either header and use a small amount of solder on each one to create a contact with the corresponding metal strip on Pico. Don't allow solder to stray across adjacent strips or pins, or you'll create a short circuit.

02 Fit the Unicorn Pack

When the solder has had time to cool, turn over Pico so that the longer end of each pin is pointing up and the USB socket is underneath. Now

▲ With the USB socket uppermost, the longer legs on your soldered headers should point downwards

check the back of the Unicorn Pack, where you'll see there's a white painted illustration of Pico's USB socket. Line up this illustration with the actual USB socket, then press the Unicorn Pack onto the header pins on Pico. Be firm but gentle, and try to keep equal pressure on either end to reduce the risk of bending any pins on either of the headers.

03 Flash Pico

Your Pico contains a web link to Raspberry Pi's own firmware, but this doesn't include the additional hooks required to work with the Unicorn Pack. Instead, you need Pimoroni's custom MicroPython firmware image. Point your browser

Press X to start a work cycle and Y to terminate the process

The LEDs are extinguished at different rates for the work and rest cycles

at **magpi.cc/pimoronipicoreleases** and download the most recent UF2 (.uf2) file – examples for use are at **magpi.cc/pimoronipicogit**. Press and hold Pico's BOOTSEL button while plugging it into a USB port on your computer (Raspberry Pi, PC, or Mac). When it appears, drag the UF2 file from your Downloads folder onto Pico. It will then reboot with the updated firmware.

" We'll code Pico using Thonny, which is included in Raspberry Pi OS "

04 Redirect Thonny

We'll code Pico using the Thonny Python IDE, which is included in Raspberry Pi OS, and free for Windows and macOS (**thonny.org**). By default, it works with and executes code stored locally, but we want to address Pico directly so that we can take advantage of the custom firmware. Click 'Python' in the lower-right corner of the Thonny window and select 'MicroPython (Raspberry Pi Pico)' from the list of options to redirect its output – you're now ready to start coding. Make sure you save your code regularly as **main.py** and, when asked where you want to save, choose 'Raspberry Pi Pico'.
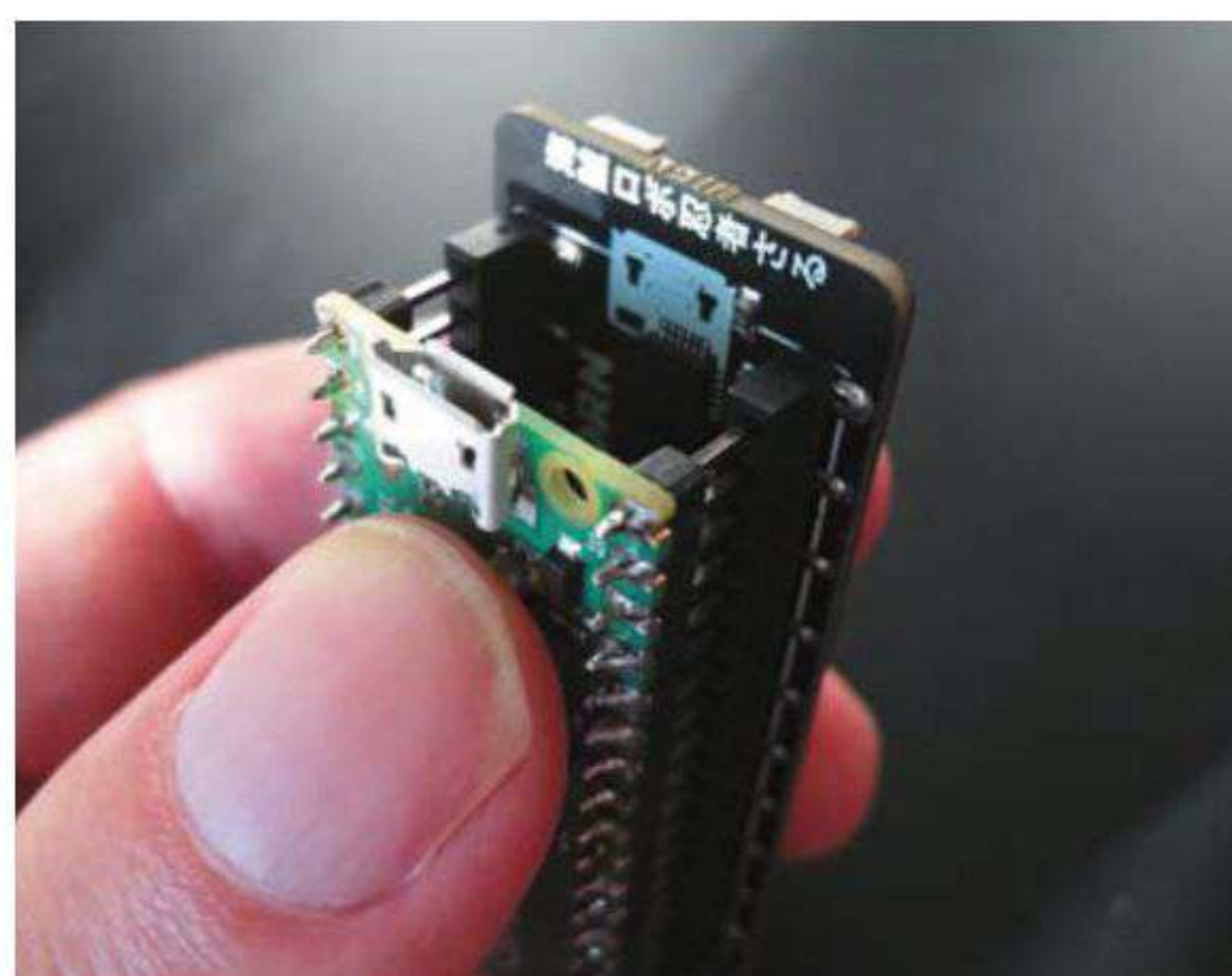
◄ When fitting the Unicorn Pack to Pico, match the rear illustration to the USB socket to make sure it's correctly orientated

05 Set up the environment

The first three lines of code set up the environment in which our program will run by referencing the libraries that handle time and the specific features of our Unicorn Pack add-on. We'll use the utime library to count the length of each work and rest cycle – which are 25 minutes (1500 seconds) and five minutes (300 seconds) respectively – and divide the number of seconds in each stretch by 112. Then, every time one 112th of a cycle has passed – 13.4 seconds on a work cycle and 2.7 on a rest cycle – we'll extinguish one of the LEDs on the Unicorn Pack.

Top Tip 👍

Why main.py?

You can save multiple apps on your Pico, but only main.py runs automatically when it's powered up.
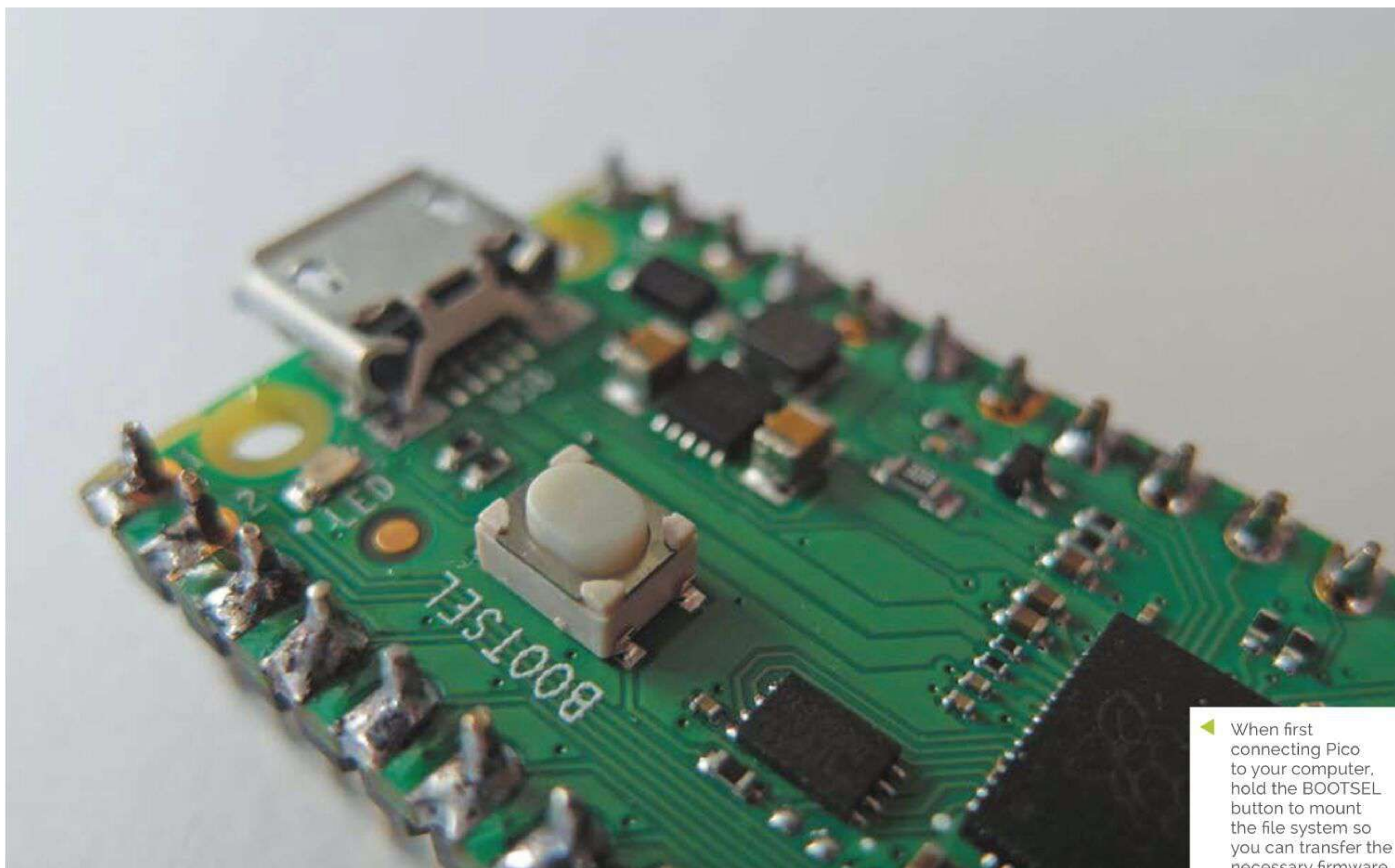
◀ When first connecting Pico to your computer, hold the BOOTSEL button to mount the file system so you can transfer the necessary firmware

⚠️ **Warning!**
Soldering

This project requires soldering. Take care when handling a hot soldering iron and solder.

magpi.cc/soldering

Top Tip 👍

Button bonanza

We're using two buttons, leaving two free. You could program one to skip to the rest cycle for comfort breaks.

06 Define a new process

We'll define a process that can be kicked into action as soon as the X button is pressed. We've called this process pomocycle, as detailed on line 7. The first job is to define the variables we'll be using, which include values for the red and green mix in the colours we want to display (red for work, green for rest), and a column and row count. Why 6 and 15 when there are 7 rows of 16 LEDs on the Unicorn Pack? Because the first row and first column of each is counted as 0, not 1.

> ❝ The important element here is the multiplier ❞

07 Build them up…

We now start a loop that keeps running as long as button Y hasn't been pressed. If it has, we exit the process and go back to waiting for a press of the X button, which starts the work and rest cycles all over again. The first job is to illuminate every LED on the Unicorn Pack, so we work through a couple of cycles, one of which is embedded within the other so they can target each pixel directly and set it to either red or green, depending on whether we're working or resting, as defined in both the opening variables and the reset cycles that appear later in the code.

08 …and knock them down

Now we start extinguishing the LEDs one by one. The important element here is the multiplier, which defines how long Pico will wait after turning off a light before targeting the next one. We could have told it to wait 13.4 seconds between each action in a work cycle, or 2.7 seconds in a rest cycle. If we did that, however, it would only be possible to interrupt a cycle by pressing Y every 13.4 or 2.7 seconds, which would both feel very unresponsive, and be extremely irritating. We need to find a way of checking the button's status during a sleep cycle.

09 Wake, sleep, repeat

The solution is to only sleep for 0.1 seconds before checking the status of the button, and to use the multiplier to define how many times the sleep/check cycle is repeated. By repeating it 134 times for each LED when working, it will take 1500.8 seconds to extinguish all of the LEDs. Conveniently, that's 0.8 seconds longer than a 25-minute work cycle. On a rest cycle, we set the multiplier to 27, so the sleep/check process loops 27 times for each LED, the last of which will then extinguish after 302.4 seconds, which is five minutes and 2.4 seconds.

▲ Make sure you select the 'MicroPython (Raspberry Pi Pico)' interpreter when working with code in Thonny

10 Subtract and start again

Every time an LED is extinguished, we subtract one from the column count and, when the next loop repeats, the LED in that spot will go out. Remember, the first column is column zero, so when the column count number hits −1 we reset the counter to 15, and instead subtract one from the row count. This continues until the row counter also hits −1, when we know there are no remaining illuminated LEDs. At this point, if we've been on a 25-minute work cycle, we need to switch to a five-minute rest cycle, and vice versa.

11 Reset the variables

If we were working, it's time to rest. So, we set the multiplier to 27 and swap the values for the red and green tones we'll use to illuminate the LEDs. Then, when the process restarts, the LEDs will shine green and blink out more quickly as they count to the end of our break. If we had been resting, the LEDs are instead set to red, and the multiplier is again set to 134 to slow down the rate at which they disappear. The code now rewinds the defined process, illuminates each LED as appropriate, and starts the next countdown.

12 Keep everything running

At this point, our code is all but complete. We just need to define what happens when we press the Y button. That's handled in lines 53 to 55, which simply extinguish any remaining LEDs on the Unicorn Pack by setting the red, green, and blue value for each one to zero. Then, the code comes to an end. With nothing else to do, Pico returns to where it started, waiting for the next time the X button is pressed, at which point the pomocycle process kicks into action once again. M

main.py

> Language: **MicroPython**

DOWNLOAD THE FULL CODE:
⬇ magpi.cc/picopomodorogit

```
001.   import math
002.   import utime
003.   import picounicorn
004.
005.   picounicorn.init()
006.
007.   def pomocycle():
008.
009.       # Set up our variables
010.       r = 255
011.       g = 0
012.       column = 15
013.       row = 6
014.       phase = "work"
015.       multiplier = 134
016.
017.       # Start counting down
018.       while not(picounicorn.is_pressed(picounicorn.BUTTON_Y)):
019.
020.           # Illuminate every LED on the board
021.           for x in range(16):
022.               for y in range(7):
023.                   picounicorn.set_pixel(x, y, r, g, 0)
024.
025.           # Extinguish LEDs one by one
026.           while row > -1:
027.               while column > -1:
028.                   for x in range(multiplier):
029.                       if not(picounicorn.is_pressed(
       picounicorn.BUTTON_Y)):
030.                           utime.sleep(0.1)
031.                       else:
032.                           break
033.                   picounicorn.set_pixel(column, row, 0, 0, 0)
034.                   column -= 1
035.               column = 15
036.               row -= 1
037.           row = 6
038.
039.           # No more LEDs? Switch from work to rest and vice versa
040.           if phase == "work":
041.               phase = "rest"
042.               multiplier = 27
043.               r = 0
044.               g = 255
045.           elif phase == "rest":
046.               phase = "work"
047.               multiplier = 134
048.               r = 255
049.               g = 0
050.           pass
051.
052.       # Clear the display
053.       for x in range(16):
054.           for y in range(7):
055.               picounicorn.set_pixel(x, y, 0, 0, 0)
056.
057.   while True:
058.       while picounicorn.is_pressed(picounicorn.BUTTON_X):
059.           pomocycle()
```

MAKER

Laura Sach

Laura leads the A Level team at the Raspberry Pi Foundation, creating resources for students to learn about Computer Science.

@CodeBoom

MAKER

Martin O'Hanlon

Martin works in the learning team at the Raspberry Pi Foundation, where he creates online courses, projects, and learning resources.

@martinohanlon

Part 03

Create GUIs with Python:
Meme Generator

Create a GUI application which draws memes

Let's take the lessons you learnt from the previous instalments to create a GUI which draws memes. You will input the text and image name and your GUI will combine them into your own meme using the Drawing widget.

Start by creating a simple GUI with two text boxes for the top and bottom text. This is where you will enter the text which will be inserted over your picture to create your meme. Add this line to import the widgets needed.

```
from guizero import App, TextBox, Drawing
```

Then add this code for the app:

```
app = App("meme")

top_text = TextBox(app, "top text")
bottom_text = TextBox(app, "bottom text")

app.display()
```

The meme will be created on a Drawing widget which will hold the image and text.

Create a meme

Add it to the GUI by inserting this code just before the `app.display()` line. The Drawing widget's height and width should be set to 'fill' the rest of the GUI.

```
meme = Drawing(app, width="fill",
height="fill")
```

The meme will be created when the text in the top and bottom text boxes changes. To do that, we will need to create a function which draws the meme.

The function should clear the drawing, create an image (we're using a photo of a woodpecker, but you can use any you want) and insert the text at the top and bottom of the image.

Remember when you used `name.value` to set the value of the Text widget with the spy name in Part 2 of this series? You can also use the value property to get the value of a Text widget, so in this case `top_text.value` means 'please get the value that is typed in the top_text box'.

```
def draw_meme():
    meme.clear()
    meme.image(0, 0, "woodpecker.png")
    meme.text(20, 20, top_text.value)
    meme.text(20, 320, bottom_text.value)
```

The first two numbers in `meme.image(0, 0)` and `meme.text(20, 20)` are the x, y co-ordinates of where to draw the image and text. The image is drawn at position `0, 0`, which is the top-left corner, so the image covers the whole of the drawing.

Finally, call your `draw_meme` function just before you display the app. Insert this code just before the `app.display` line:

```
draw_meme()
```

Your code should now look like **meme1.py**. If you run your app (**Figure 1**) and try changing

Figure 1 Meme with unstyled text

the top and bottom text, you will notice that it doesn't update in the meme. To get this working, you will have to change your program to call the draw_meme function when the text changes, by adding a command to the two TextBox widgets to the app.

```
  top_text = TextBox(app, "top text",
command=draw_meme)
  bottom_text = TextBox(app, "bottom text",
command=draw_meme)
```

❝ Update your meme by changing the top and bottom text ❞

Your code should now look like that in **meme2.py**. Run it and update your meme by changing the top and bottom text.

You can alter the look of your meme by changing the color, size, and font parameters of the text. For example:

```
meme.text(
    20, 20, top_text.value,
    color="orange",
    size=40,
    font="courier")
meme.text(
    20, 320, bottom_text.value,
```

meme1.py

DOWNLOAD THE FULL CODE:

⬇ magpi.cc/guizerocode

❯ Language: **Python 3**

```
001.   # Imports ---------------
002.
003.   from guizero import App, TextBox, Drawing
004.
005.
006.   # Functions -------------
007.
008.   def draw_meme():
009.       meme.clear()
010.       meme.image(0, 0, "woodpecker.png")
011.       meme.text(20, 20, top_text.value)
012.       meme.text(20, 320, bottom_text.value)
013.
014.
015.   # App -------------------
016.
017.   app = App("meme")
018.
019.   top_text = TextBox(app, "top text")
020.   bottom_text = TextBox(app, "bottom text")
021.
022.   meme = Drawing(app, width="fill", height="fill")
023.
024.   draw_meme()
025.
026.   app.display()
```

meme2.py

❯ Language: **Python 3**

```
001.   # Imports ---------------
002.
003.   from guizero import App, TextBox, Drawing
004.
005.
006.   # Functions -------------
007.
008.   def draw_meme():
009.       meme.clear()
010.       meme.image(0, 0, "woodpecker.png")
011.       meme.text(20, 20, top_text.value)
012.       meme.text(20, 320, bottom_text.value)
013.
014.
015.   # App -------------------
016.
017.   app = App("meme")
018.
019.   top_text = TextBox(app, "top text", command=draw_meme)
020.   bottom_text = TextBox(app, "bottom text", command=draw_meme)
021.
022.   meme = Drawing(app, width="fill", height="fill")
023.
024.   draw_meme()
025.
026.   app.display()
```

```
        color="blue",
        size=28,
        font="times new roman",
        )
```

Your code should now look like **meme3.py**. Try different styles until you find something you like (**Figure 2**).

Customise your meme generator

For a truly interactive meme generator, the user should be able to set the font, size, and colour themselves. You can provide additional widgets on the GUI to allow them to do this.

meme3.py

> Language: **Python 3**

```
001.   # Imports ---------------
002.
003.   from guizero import App, TextBox, Drawing
004.
005.
006.   # Functions -------------
007.
008.   def draw_meme():
009.       meme.clear()
010.       meme.image(0, 0, "woodpecker.png")
011.       meme.text(
012.           20, 20, top_text.value,
013.           color="orange",
014.           size=40,
015.           font="courier")
016.       meme.text(
017.           20, 320, bottom_text.value,
018.           color="blue",
019.           size=28,
020.           font="times new roman",
021.           )
022.
023.
024.   # App -------------------
025.
026.   app = App("meme")
027.
028.   top_text = TextBox(app, "top text", command=draw_meme)
029.   bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.   meme = Drawing(app, width="fill", height="fill")
032.
033.   draw_meme()
034.
035.   app.display()
```

The number of options available for the colour and font are limited, so you could use a drop-down list, also known as a Combo, for this. The size could be set using a Slider widget.

First, modify your import statement to include the Combo and Slider widgets.

```
from guizero import App, TextBox, Drawing, Combo, Slider
```

After you have created your TextBox widgets for the top and bottom text, create a new Combo widget so the user can select a colour.

```
bottom_text = TextBox(app, "bottom text",
command=draw_meme)
color = Combo(app,
    options=["black", "white", "red",
"green", "blue", "orange"],
    command=draw_meme)
```

The `options` parameter sets what colours the user can select from the Combo. Each colour is an element in a list. You can add any other colours you want to the list.

> ❝ The options are displayed in the order in which you put them in the list ❞

The options are displayed in the order in which you put them in the list. The first option is the default, which is displayed first. If you want to have a different option as the default, you can do it using the `selected` parameter, e.g. `"blue"`.

```
color = Combo(app,
    options=["black", "white", "red",
"green", "blue", "orange"],
    command=draw_meme,
    selected="blue")
```

Now your user can select a colour. Next, you need to change the `draw_meme` function to use Combo's value when creating the text in your the meme. For example:
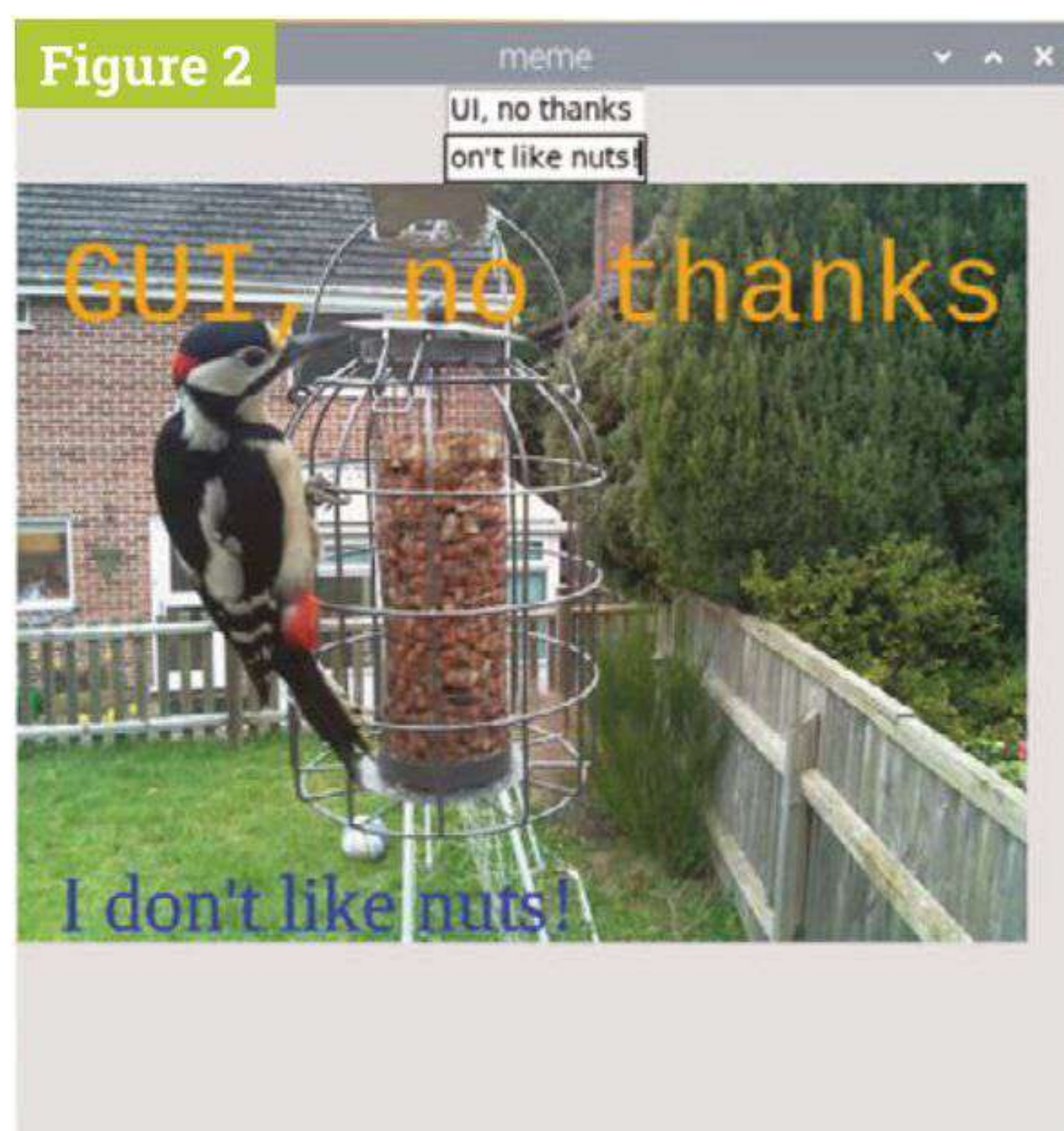
Figure 2

▲ **Figure 2** Alter the fonts and colours

```
meme.text(
    20, 20, top_text.value,
    color=color.value,
    size=40,
    font="courier")
```

Do the same for the bottom-text block of code. Your program should now resemble **meme4.py**.

Following the steps above, add a second Combo to your application so the user can select a font from this list of options: ["times new roman", "verdana", "courier", "impact"]. Remember to change the draw_meme function to use the font value when adding the text.

Create a new Slider widget to set the size of the text your user wants.

```
size = Slider(app, start=20, end=40,
command=draw_meme)
```

The range of the slider is set using the start and end parameters. So, in this example, the smallest text available will be 20 and the largest 40.

Modify the draw_meme function to use the value from your size slider when creating the meme's text.

```
meme.text(
    20, 20, top_text.value,
```

meme4.py

> Language: **Python 3**

```
001.    # Imports ---------------
002.
003.    from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.    # Functions -------------
007.
008.    def draw_meme():
009.        meme.clear()
010.        meme.image(0, 0, "woodpecker.png")
011.        meme.text(
012.            20, 20, top_text.value,
013.            color=color.value,
014.            size=40,
015.            font="courier")
016.        meme.text(
017.            20, 320, bottom_text.value,
018.            color=color.value,
019.            size=28,
020.            font="times new roman",
021.            )
022.
023.
024.    # App -------------------
025.
026.    app = App("meme")
027.
028.    top_text = TextBox(app, "top text", command=draw_meme)
029.    bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.    color = Combo(app,
032.                  options=["black", "white", "red", "green",
"blue", "orange"],
033.                  command=draw_meme, selected="blue")
034.
035.    meme = Drawing(app, width="fill", height="fill")
036.
037.    draw_meme()
038.
039.    app.display()
```

Drawing widget

The Drawing widget is really versatile and can be used to display lots of different shapes, patterns, and images. To find out more about the Drawing widget, see Appendix C of the book (**magpi.cc/pythongui**), or take a look at the online documentation: **lawsie.github.io/guizero/drawing**.

04-meme-generator.py

> Language: **Python 3**

```
001.    # Imports ---------------
002.
003.    from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.    # Functions ------------
007.
008.    def draw_meme():
009.        meme.clear()
010.        meme.image(0, 0, "woodpecker.png")
011.        meme.text(
012.            20, 20, top_text.value,
013.            color=color.value,
014.            size=size.value,
015.            font=font.value)
016.        meme.text(
017.            20, 320, bottom_text.value,
018.            color=color.value,
019.            size=size.value,
020.            font=font.value,
021.            )
022.
023.
024.    # App -------------------
025.
026.    app = App("meme")
027.
028.    top_text = TextBox(app, "top text", command=draw_meme)
029.    bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.    color = Combo(app,
032.            options=["black", "white", "red", "green",
    "blue", "orange"],
033.            command=draw_meme, selected="blue")
034.
035.    font = Combo(app,
036.            options=["times new roman", "verdana", "courier",
    "impact"],
037.            command=draw_meme)
038.
039.    size = Slider(app, start=20, end=50, command=draw_meme)
040.
041.    meme = Drawing(app, width="fill", height="fill")
042.
043.    draw_meme()
044.
045.    app.display()
```
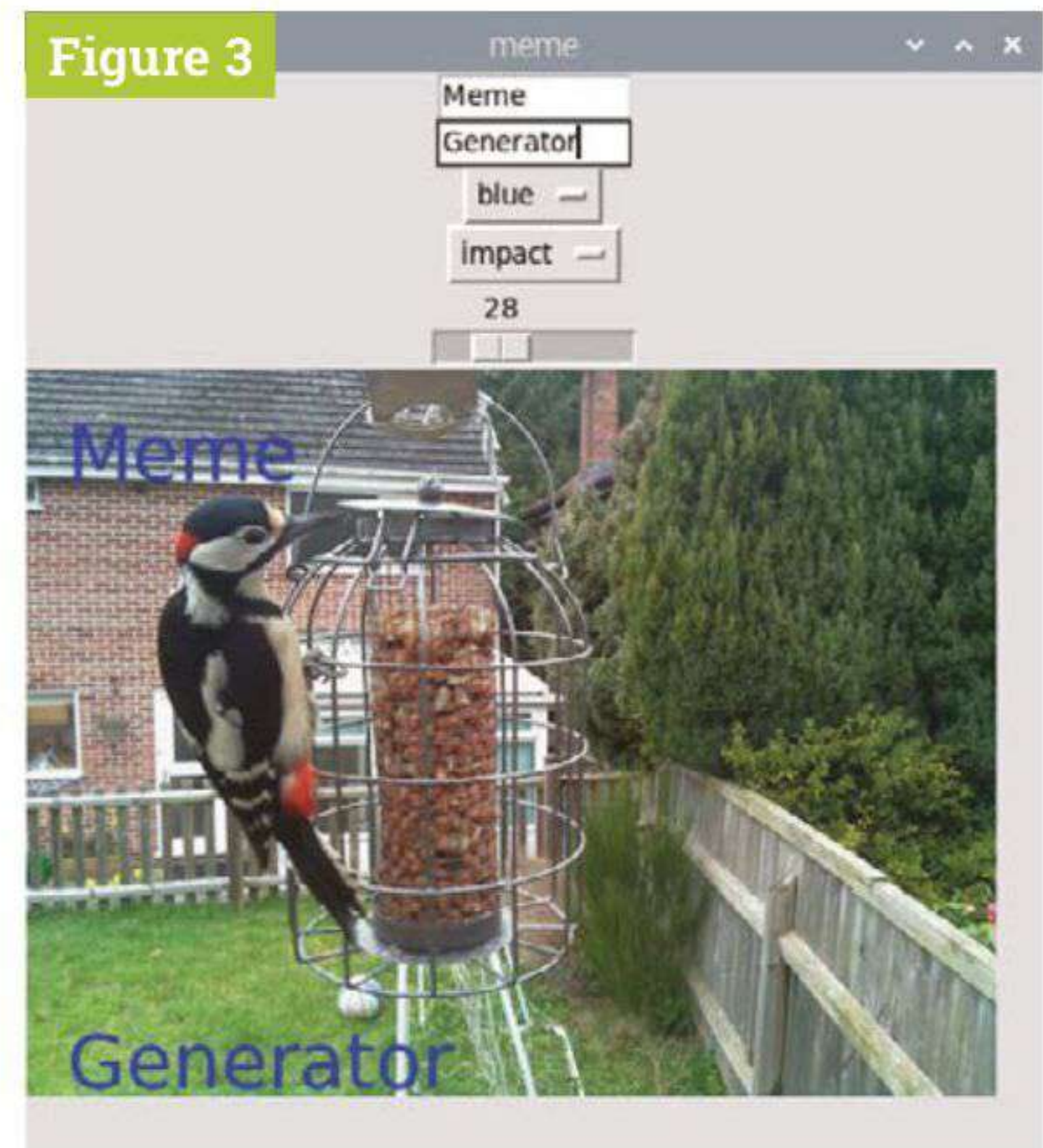

Figure 3 The finished meme generator

```
        color=color.value,
        size=size.value,
        font=font.value)
```

Your code should now resemble that in **04-meme-generator.py**. Try running it and you should see something like **Figure 3**.

Can you change the GUI so that the name of the image file can be entered into a TextBox or perhaps selected from a list in a Combo? This would make your application capable of generating memes with different images too. **M**

Create Graphical User Interfaces with Python

For further tutorials on how to make your own GUIs with guizero, take a look at our book, *Create Graphical User Interfaces with Python*. Its 156 pages are packed with essential info and a range of exciting projects.
magpi.cc/pythongui
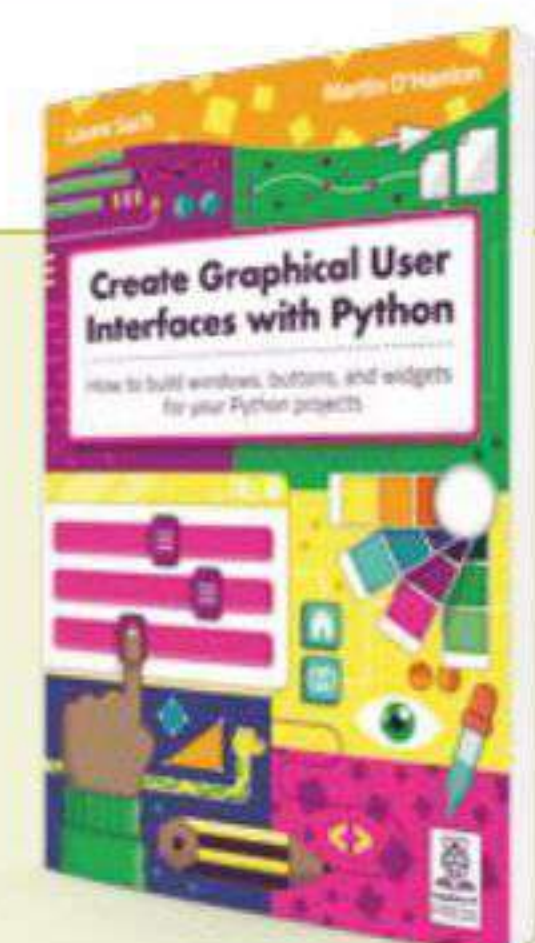
Make a digital
do-not-disturb sign

Kids barging in on Zoom meetings? Parents being a pain when you're in Google Classroom? Let them know what you're up to with 'Busybot', the custom do-not-disturb sign

MAKER

PJ Evans

PJ is a writer, tinkerer, and software engineer. He spends all day on Zoom and no longer knows if his colleagues have legs or not.

@mrpjevans

There are often times when family and work life collide. The internet is littered with videos of online meetings being hijacked by partners, parents, and pets. We can't do anything about your cat, but maybe a digital sign can let people know what you're up to, and whether it's OK to wander in.

In this tutorial we're going to use a popular messaging protocol, MQTT (Message Queuing Telemetry Transport). With MQTT, we will set up a simple do-not-disturb sign that you can trigger with a single key press and customise to your heart's content.
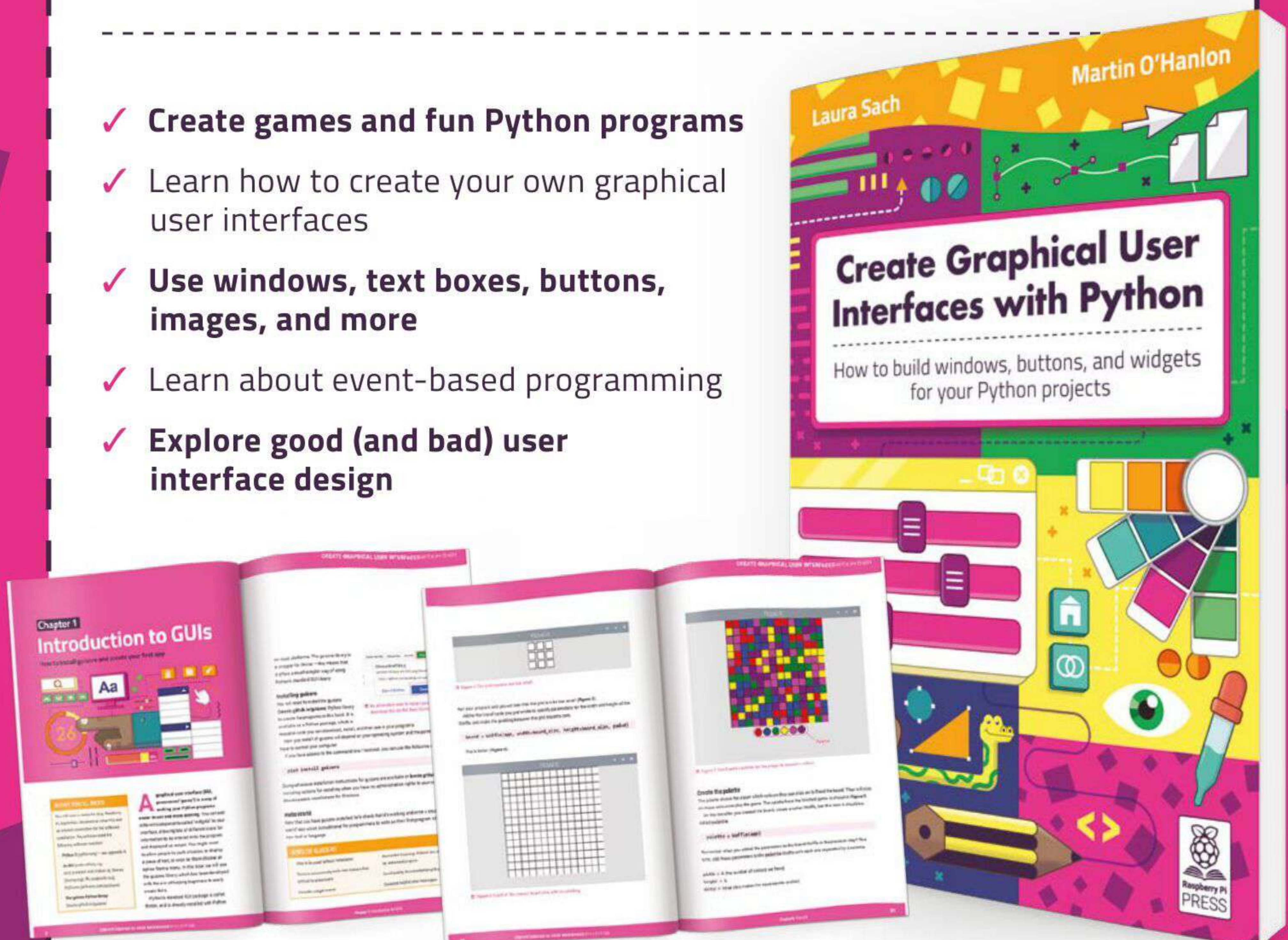
01 Prepare Raspberry Pi

We're using two Raspberry Pi Zero W computers for this project, although it will work with any recent model. For both, we recommend installing Raspberry Pi OS Lite (**magpi.cc/software**) as we don't need a desktop interface. Make sure both devices are working and connected to the same network. We also recommend setting their host names by running `sudo raspi-config` at the command line, then going to System Options > Hostname. We chose 'busybot' for the sign and 'buttonbot' for the controller, and that's how we'll refer to them throughout this tutorial. Finally, make sure everything is up to date. Enter these commands:

```
sudo apt update
sudo apt full-upgrade
```

You should be able to ping one device from the other. On 'busybot' Raspberry Pi Zero W, enter:

```
ping buttonbot.local
```

You should get a response from 'buttonbot' Raspberry Pi Zero W.

You'll Need

> 2 × Raspberry Pi Zero W
> **magpi.cc/pizerow**

> Scroll pHAT HD
> **magpi.cc/ scrollhathd**

> pHAT Diffuser
> **magpi.cc/ phatdiffuser**

> Keybow Kit (3-key)
> **magpi.cc/keybow3**

02 Install the Scroll pHAT HD and diffuser

We're going to set up the display first. Pimoroni's Scroll pHAT HD is a display made up of 17×7 pixels (119 total) and comes with a Python library that does all the hard work of displaying and scrolling text for us. The pHAT form factor makes it just the right size for a display, but of course you can adapt this tutorial to any display you like. You'll need to solder the 40-pin header to the display and also add a reciprocal header to 'busybot' Raspberry Pi if it doesn't already have one. Before assembling them together, screw on the diffuser, which will make the display much easier to read. Now, with Raspberry Pi powered off, carefully connect the Scroll pHAT HD.

03 Set up the display software

It's time to make sure our display is working before we go any further. Fortunately, Pimoroni has created an install script for us. At the command line, run the following:

```
curl https://get.pimoroni.com/scrollphathd | bash
```

Make sure you do the 'Full Install' when the option is presented. This will set up Raspberry Pi so it can communicate with the display and install all the Python libraries we need. Take a look at Pimoroni's GitHub (**magpi.cc/scrollphatgit**) for more information on installation.
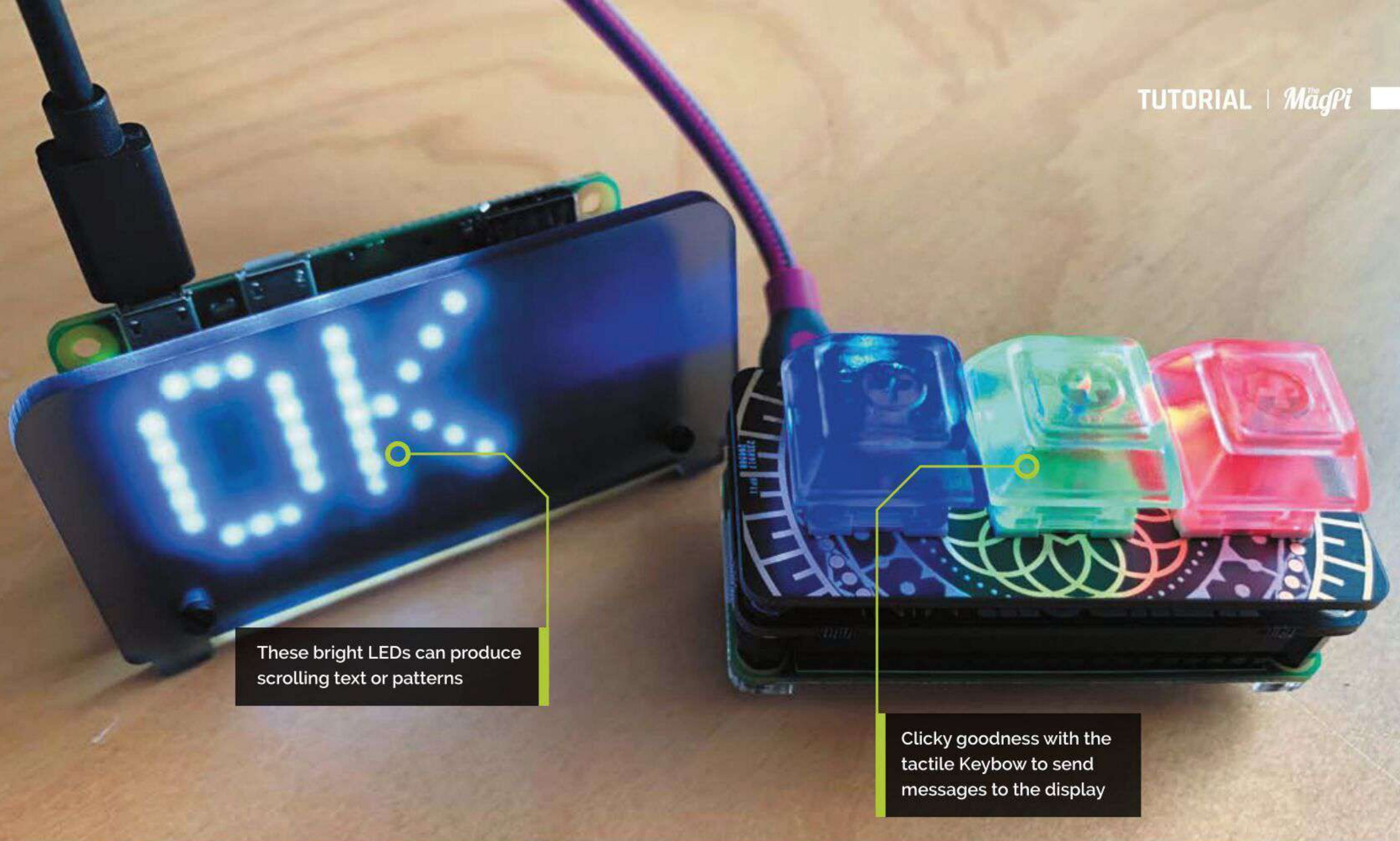
Once the install has finished, we can test things out. Reboot, then try this on the command line:

```
cd ~/Pimoroni/scrollphathd/examples
python3 swirl.py
```

See a pretty pattern? Then you're ready to proceed. Have fun with the other examples in the directory; they're a great source of inspiration.

These bright LEDs can produce scrolling text or patterns

Clicky goodness with the tactile Keybow to send messages to the display

04 The interface

Our buttonbot and busybot are going to need to talk to each other over the network. One of the easiest, and most popular, ways to do this is the MQTT protocol. It uses a pub/sub model (publisher / subscriber) to process messages. An MQTT server (the 'broker') receives messages from 'publishers' that are then transmitted to 'subscribers'. These are organised by 'topic'. The biggest advantage is that publishers don't need to understand or even be aware of the subscribers, they just need to speak MQTT. Don't worry if this is confusing; working through the tutorial will make things clearer.

05 Mosquitto

For our system to work, you need an MQTT server (or 'broker') to handle the messages. This can be anywhere on your network, but for the purposes of the tutorial we'll install it on the same 'busybot' Raspberry Pi driving the display. MQTT software tends to be very 'lightweight', so a Raspberry Pi Zero W can easily handle being the server as well as a publisher. Mosquitto is probably the most popular set of MQTT tools. Installation is straightforward, too. Enter this at the command line:

```
sudo apt install mosquitto mosquitto-clients
sudo pip3 install paho-mqtt
```

The broker will be automatically installed as a service and always be running in the background. We've also installed Paho, a popular Python library for implementing MQTT.

06 Connectors

The code for this project does two main jobs: listens to the MQTT server for new messages, and then takes those messages and scrolls them on the display. You can enter the code shown here or get the files from **magpi.cc/busybotgit**. We need the code to be running all the time. To do this, create a new file from the command line:

```
sudo nano /usr/lib/systemd/busybot.service
```

▼ The MQTT protocol is perfect for Internet of Things projects

Publisher

"I'm on Zoom" to 'busybot' topic

MQTT Broker

"I'm on Zoom"   "I'm on Zoom"   "I'm on Zoom"

"I'm on Zoom"

Subscriber to 'busybot' topic

Subscriber to 'busybot' topic

Subscriber to 'busybot' topic
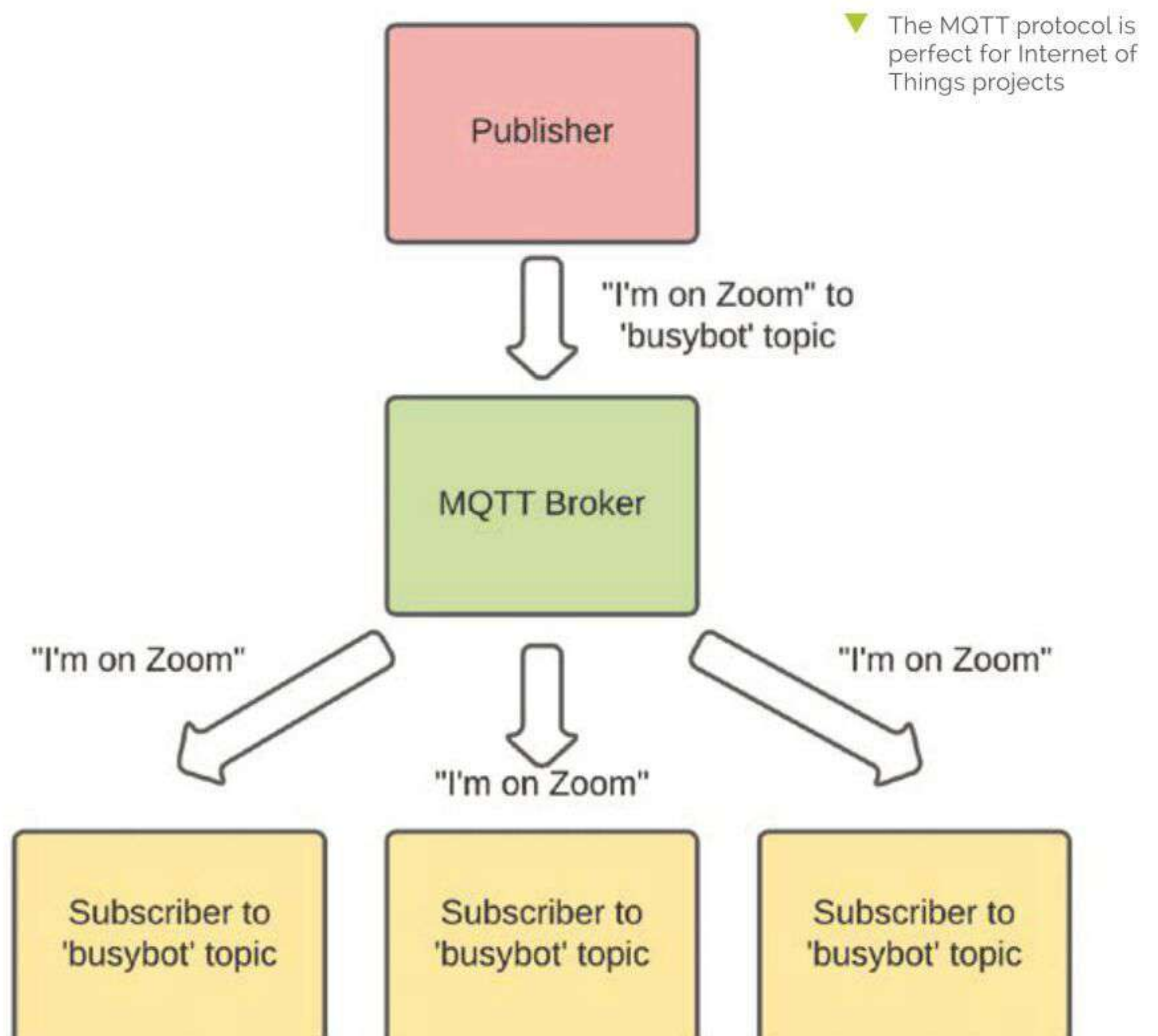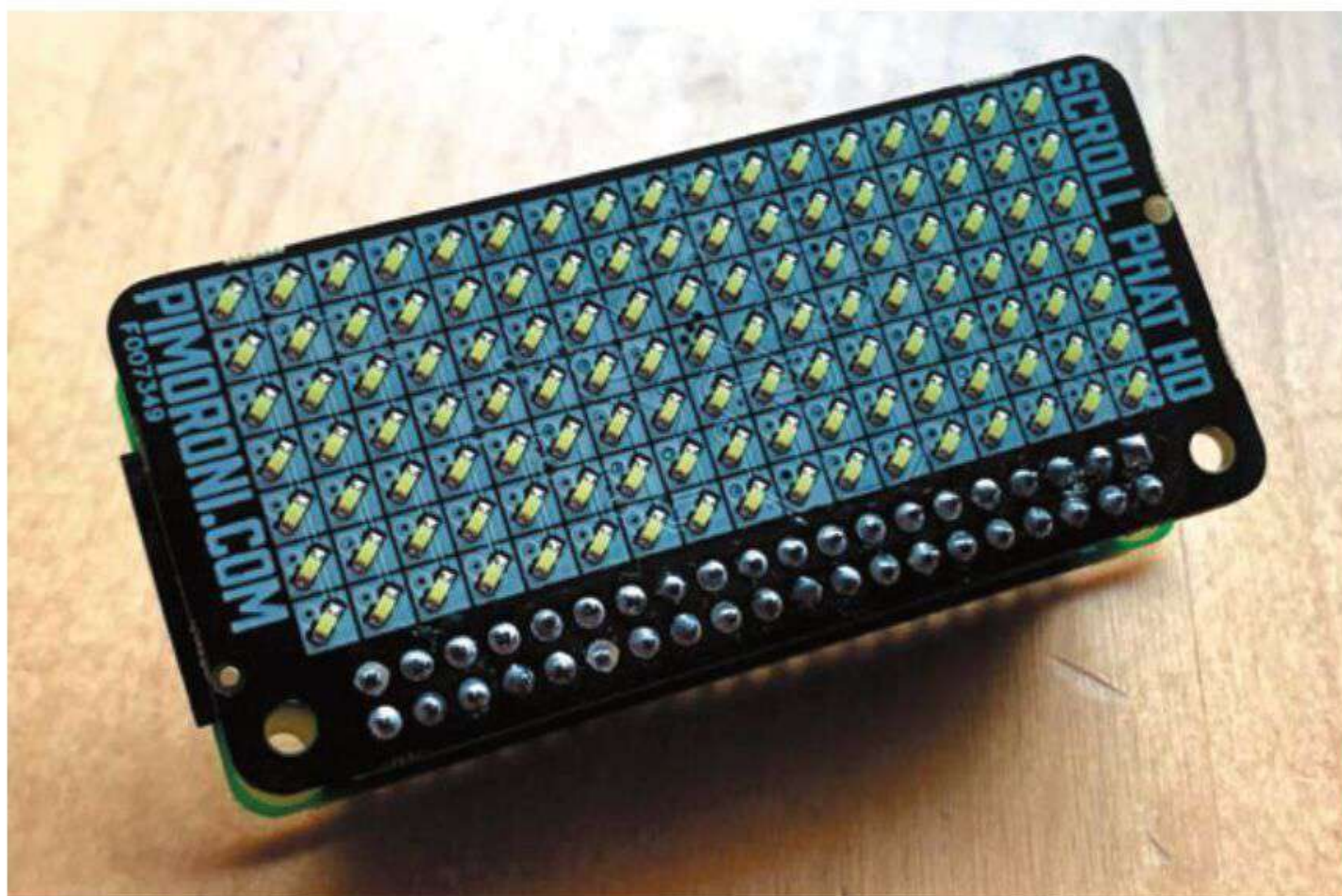
▲ The Scroll pHAT HD without its diffuser. 119 very bright LEDs

▲ Need more buttons? Try the nine-key Keybow

Enter the code from **busybot.service**. (Change the paths if you've created the code elsewhere.) Save the file (**CTRL+X** and then **Y**), then enter these commands:

```
sudo systemctl enable /usr/lib/systemd/
busybot.service
  sudo systemctl start busybot
```

Top Tip 👍

In the upside-down?

If your display is looking a little topsy-turvy, you can flip the display by uncommenting the line containing `scrollphathd. flip(1, 1)` in **busybot_ magpi.py**.

07 Testing time

Let's confirm that the code is working. Using the Paho MQTT libraries, the code subscribes to the MQTT topic 'busybot' on the broker. Whenever anything publishes a line of text to that topic, busybot will be notified, the text delivered and then displayed. We can check everything is working using the Mosquitto command line publishing tool:

```
mosquitto_pub -h localhost -t busybot -m
"Hello from MagPi"
```

If you see the messages scrolling across, then everything is working. Send any message you wish, or a blank space to clear the display.

08 Assemble the Keybow

Now we have our display working, it's time to turn our attention to sending the messages. MQTT is widely supported and you can get clients for almost every platform and programming language in common use. That means we can send messages to the display from pretty much anywhere. In this tutorial, we're going to use Pimoroni's Keybow interface to provide a quick way of setting messages. On the second 'buttonbot' Raspberry Pi WH, assemble the Keybow as instructed at **magpi.cc/assemblekeybowmini**, but don't install Keybow OS – we'll stick with Raspberry Pi OS Lite.

> ❝ Whenever anything publishes a line of text to that topic, busybot will be notified ❞

09 Keybow setup

We'll now add some code to send messages to the MQTT broker when buttons are pressed. First, from the command line, we need to install some dependencies on buttonbot:

```
sudo apt install python3-pip git
sudo pip3 install keybow paho-mqtt
```

Now get the code from GitHub:

```
cd
git clone https://github.com/mrpjevans/
busybot.git
```

We can now test the Keybow with a simple example:

busybot.service

> Language: **Bash**

```
001.  [Unit]
002.  Description=busybot
003.
004.  [Service]
005.  ExecStart=/usr/bin/python3 /home/pi/busybot/busybot_magpi.py
006.  Restart=on-failure
007.  User=pi
008.  Group=pi
009.
010.  [Install]
011.  WantedBy=multi-user.target
```

```
cd ~/busybot
python3 test_keybow.py
```

Press the keys. Do they all light up? Then all is well.

10 Keybow code

In the **busybot** directory, have a look at **buttonbot.py**. You'll also need to change the name of the MQTT broker and/or topic if you've used something different.

We need to make sure the code is always running, just like busybot. Again, we'll create a service to do this. From buttonbot's command line, go through the process in Step 6 to create a service file and enable it. Just make sure you change the **ExecStart** line to:

```
ExecStart=/usr/bin/python3 /home/pi/busybot/
buttonbot.py
```

Save the file and enable it as before.

11 Testing and tinkering

Everything should now be ready. With both Raspberry Pi Zero computers running, try pressing a key on the Keybow. A message will now scroll across the display on the other Raspberry Pi Zero. Make sure all three keys work. You're now ready to start customising the system to your own needs. If you edit **buttonbot.py**, you'll see some documented options for changing the messages and the colours of the keys. Free free to experiment, make changes, and make this code your own. If you need more than three messages, see if you can alter the code to support key press combinations, which would give you up to seven options.

12 Python vs C++

As the display doesn't 'know' about buttonbot's existence, it means that anything capable of speaking MQTT can send messages to busybot to set the scrolling text. It could be done from the command line or when a certain event happens. If you've been following the Home Assistant (HA) series in *The MagPi*, then you may be interested to know that HA speaks MQTT, so the display could be tied to a temperature/motion sensor. Have fun dreaming up new ideas. 🎵

busybot_magpi.py

> Language: **Python 3**

DOWNLOAD THE FULL CODE:
magpi.cc/busybotgit

```
001.  import time
002.  import scrollphathd
003.  import paho.mqtt.client as mqtt
004.
005.  # Change these to suit your needs
006.  broker = '192.168.0.100'
007.  client_name = 'busybot'
008.  topic = 'study/busybot'
009.  brightness = 0.1
010.
011.  current_message = ''
012.
013.
014.  def scroll_message():
015.      # Original function by Pimoroni x
016.      global current_message
017.      while True:
018.
019.          # No message? Don't do anything.
020.          if len(current_message) is 0:
021.              scrollphathd.clear()
022.              time.sleep(1)
023.              continue
024.
025.          # Clear the display and reset scrolling to (0, 0)
026.          scrollphathd.clear()
027.          length = scrollphathd.write_string(current_message)
028.          scrollphathd.show()
029.          time.sleep(0.5)
030.
031.          length -= scrollphathd.width
032.
033.          # Now for the scrolling loop...
034.          while length > 0:
035.              # Scroll the buffer one place to the left
036.              scrollphathd.scroll(1)
037.              scrollphathd.show()
038.              length -= 1
039.              time.sleep(0.02)
040.
041.          time.sleep(0.5)
042.
043.
044.  def on_message(client, userdata, message):
045.      global current_message
046.      print('MQTT Message received')
047.      current_message = message.payload.decode("utf-8")
048.      if len(current_message) == 0:
049.          print('Clearing message')
050.      else:
051.          print('Scrolling ' + current_message)
052.
053.
054.  print('Starting')
055.  client = mqtt.Client(client_name)
056.  client.connect(broker)
057.  client.subscribe(topic)
058.  client.on_message = on_message
059.
060.  print('Listening to ' + topic)
061.  client.loop_start()
062.
063.  scrollphathd.set_brightness(brightness)
064.  scroll_message()
```
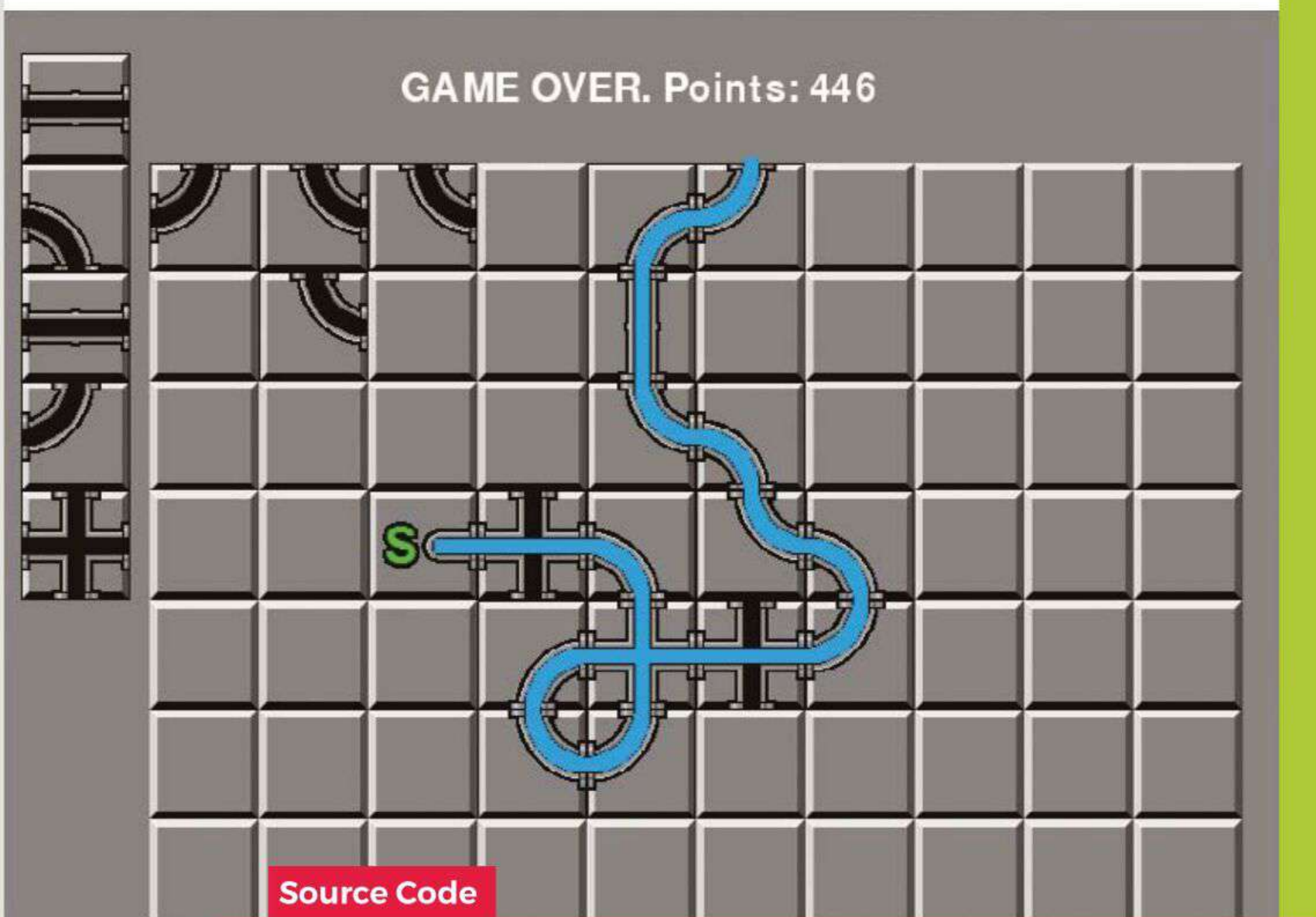
GAME OVER. Points: 446

Code your own
Pipe Mania puzzler

Create a network of pipes before the water starts to flow in our re-creation of a classic puzzler

AUTHOR
JORDI SANTONJA

Pipe Mania, also called *Pipe Dream* in the US, is a puzzle game developed by The Assembly Line in 1989 for Amiga, Atari ST, and PC, and later ported to other platforms, including arcades. The player must place randomly generated sections of pipe onto a grid. When a counter reaches zero, water starts to flow and must reach the longest possible distance through the connected pipes.

Let's look at how to recreate *Pipe Dream* in Python and Pygame Zero. The variable **start** is decremented at each frame. It begins with a value of **60*30**, so it reaches zero after 30 seconds if our monitor runs at 60 frames per second. In that time, the player can place tiles on the grid to build a path. Every time the user clicks on the grid, the last tile from **nextTiles** is placed on the play area and a new random tile appears at the top of the next tiles. **randint(2,8)** computes a random value between 2 and 8.

grid and **nextTiles** are lists of tile values, from 0 to 8, and are copied to the screen in the **draw** function with the **screen.blit** operation. **grid** is a two-dimensional list, with sizes **gridWidth=10** and **gridHeight=7**. Every pipe piece is placed in **grid** with a mouse click. This is managed with the Pygame functions **on_mouse_move** and **on_mouse_down**, where the variable **pos** contains the mouse position in the window. **panelPosition** defines the position of the top-left corner of the grid in the window. To get the grid cell, **panelPosition** is subtracted from **pos**, and the result is divided by **tileSize** with the integer division **//**. **tileMouse** stores the resulting cell element, but it is set to **(-1,-1)** when the mouse lies outside the grid.

The **images** folder contains the PNGs with the tile images, two for every tile: the graphical image and the path image. The **tiles** list contains the name of every tile, and adding to it **_block** or **_path** obtains the name of the file. The values stored in **nextTiles** and **grid** are the indexes of the elements in **tiles**.

The image **waterPath** isn't shown to the user, but it stores the paths that the water is going to follow. The first point of the water path is located in the starting tile, and it's stored in **currentPoint**. **update** calls the function **CheckNextPointDeleteCurrent**, when the water starts flowing. That function finds the next point in the water path, erases it, and adds a new point to the **waterFlow** list. **waterFlow** is shown to the user in the **draw** function.

pointsToCheck contains a list of relative positions, offsets, that define a step of two pixels from **currentPoint** in every direction to find the next point. Why two pixels? To be able to define the 'cross' tile, where two lines cross each other. In a 'cross' tile the water flow must follow a straight line, and this is how the only points found are the next points in the same direction. When no next point is found, the game ends and the score is shown: the number of points in the water path, **playState** is set to **0**, and no more updates are done. Ⓦ

Pipe-wrangling in Python

Here's Jordi's code for a *Pipe Mania*-style puzzler. To get it working on your system, you'll need to install Pygame Zero – full instructions are available at **wfmag.cc/pgzero**.

```python
# Pipe Mania
from pygame import image, Color, Surface
from random import randint

gridWidth, gridHeight = 10, 7
grid = [[0 for x in range(gridWidth)] for y in range(gridHeight)]
tileSize = 68
panelPosition = (96, 96)
numberNextTiles = 5
nextTiles = [randint(2, 8) for y in range(numberNextTiles)]
nextTilesPosition = (16, 28)
tileMouse = (-1, -1)

tiles = ['empty', 'start',
        'hori', 'vert', 'cross',
        'bottomleft', 'bottomright',
        'topleft', 'topright']

pathTiles = [image.load('images/'+tiles[i]+'_path.png') for i in
range(1,9)]

waterPath = Surface((gridWidth*tileSize, gridHeight*tileSize))
waterPath.fill(Color('black'))
grid[3][2] = 1 # start tile
waterPath.blit(pathTiles[0], (2 * tileSize, 3 * tileSize))
currentPoint = (2 * tileSize + 43, 3 * tileSize + 34)
waterFlow = []
start = 60*30 # 30 seconds

playState = 1

pointsToCheck = [(2, 0),( 0,2),(-2, 0),( 0,-2),
                (2, 1),( 1,2),(-2, 1),( 1,-2),
                (2,-1),(-1,2),(-2,-1),(-1,-2),
                (2,-2),( 2,2),(-2, 2),(-2,-2)]

def draw():
    screen.blit('background', (0,0))
    for x in range(gridWidth):
        for y in range(gridHeight):
            screen.blit(tiles[grid[y][x]]+'_block', (
                panelPosition[0] + x * tileSize,
                panelPosition[1] + y * tileSize))
    for y in range(numberNextTiles):
        screen.blit(tiles[nextTiles[y]]+'_block', (
            nextTilesPosition[0],
            nextTilesPosition[1] + y * tileSize))
    for point in waterFlow:
        screen.blit('water', point)
    if playState == 1:
        if tileMouse[0] >= 0 and tileMouse[1] >= 0:
            screen.blit(tiles[nextTiles[-1]]+'_block', (
                panelPosition[0] + tileMouse[0] * tileSize,
                panelPosition[1] + tileMouse[1] * tileSize))
```

```python
        if start > 0:
            screen.draw.text("Start in "
+ str(start // 60), center=(400, 50), fontsize=35)
        else:
            screen.draw.text("GAME OVER. Points:
"+str(len(waterFlow)), center=(400, 50), fontsize=35)

def update():
    global start, playState
    if start > 0:
        start -= 1
    elif playState == 1:
        if not CheckNextPointDeleteCurrent():
            playState = 0

def CheckNextPointDeleteCurrent():
    global currentPoint
    for point in pointsToCheck:
        newPoint = (currentPoint[0] + point[0], currentPoint[1]
+ point[1])
        if newPoint[0] < 0 or newPoint[1] < 0 or newPoint[0] >=
gridWidth*tileSize or newPoint[1] >= gridHeight*tileSize:
            return False # goes outside the screen
        if waterPath.get_at(newPoint) != Color('black'):
            waterPath.set_at(newPoint, Color('black'))
            waterFlow.append((newPoint[0] + panelPosition[0] - 4,
newPoint[1] + panelPosition[1] - 4))
            currentPoint = newPoint
            return True
    return False # no next point found


def on_mouse_down(pos):
    if playState == 1 and tileMouse[0] >= 0 and tileMouse[1] >=
0:
        if grid[tileMouse[1]][tileMouse[0]] != 1: # not start
tile
            grid[tileMouse[1]][tileMouse[0]] = nextTiles[-1]
            waterPath.fill(Color('black'), (tileMouse[0] *
tileSize, tileMouse[1] * tileSize, tileSize, tileSize))
            waterPath.blit(pathTiles[nextTiles[-1] - 1],
(tileMouse[0] * tileSize, tileMouse[1] * tileSize))
            for i in reversed(range(numberNextTiles - 1)):
                nextTiles[i + 1] = nextTiles[i]
            nextTiles[0] = randint(2, 8)

def on_mouse_move(pos):
    global tileMouse
    if playState == 1:
        tileMouse = ((pos[0] - panelPosition[0])//tileSize,
(pos[1] - panelPosition[1])//tileSize)
        if pos[0] < panelPosition[0] or pos[1] < panelPosition[1]
or tileMouse[0] >= gridWidth or tileMouse[1] >= gridHeight:
            tileMouse = (-1, -1) # mouse outside panel
```

Turbocharge Raspberry Pi 400 with an M.2 SATA SSD drive

Boost your storage speeds with a super-fast M.2 SSD and learn how to adjust boot order in Raspberry Pi OS

MAKER

Lucy Hattersley

Lucy is editor of *The MagPi* and loves her Raspberry Pi 400 as well as her 8GB Raspberry Pi 4. Sometimes it's hard to pick a favourite.

magpi.cc

Recently we looked at a superb case from Argon (magpi.cc/argononem2) which transformed our Raspberry Pi 4 by upgrading the boot drive to an M.2 SSD.

The result was a tenfold increase in storage speed, making for faster performance across the board. Apps load more quickly, and browsing the internet is vastly improved. M.2 SATA is also great for working with large, demanding files such as video, large photo images, and big data files.

The latest offering from Raspberry Pi and our favourite all-in-one computer is Raspberry Pi 400.

So we set about sourcing a compatible solution for Raspberry Pi 400. Thanks to the USB 3.0 ports on the rear of Raspberry Pi 400, and recent default support for USB boot, it turns out to be easy to upgrade a Raspberry Pi 400 in the same manner.

All you need to do is source a compatible M.2 SATA drive and M.2 SATA to USB 3.0 enclosure. Put the two together and hook the unit up to Raspberry Pi 400, then copy across the operating system and you're good to boot.

We used a Transcend M.2 SSD 430S (**magpi.cc/430s**) and Transcend TSCM42S USB

You'll Need

> M.2 SSD
> **magpi.cc/430s**

> M.2 SATA to USB 3.1 SSD Enclosure Kit (TSCM42S)
> **magpi.cc/tscm42**

> Raspberry Pi 400
> **magpi.cc/ raspberrypi400**
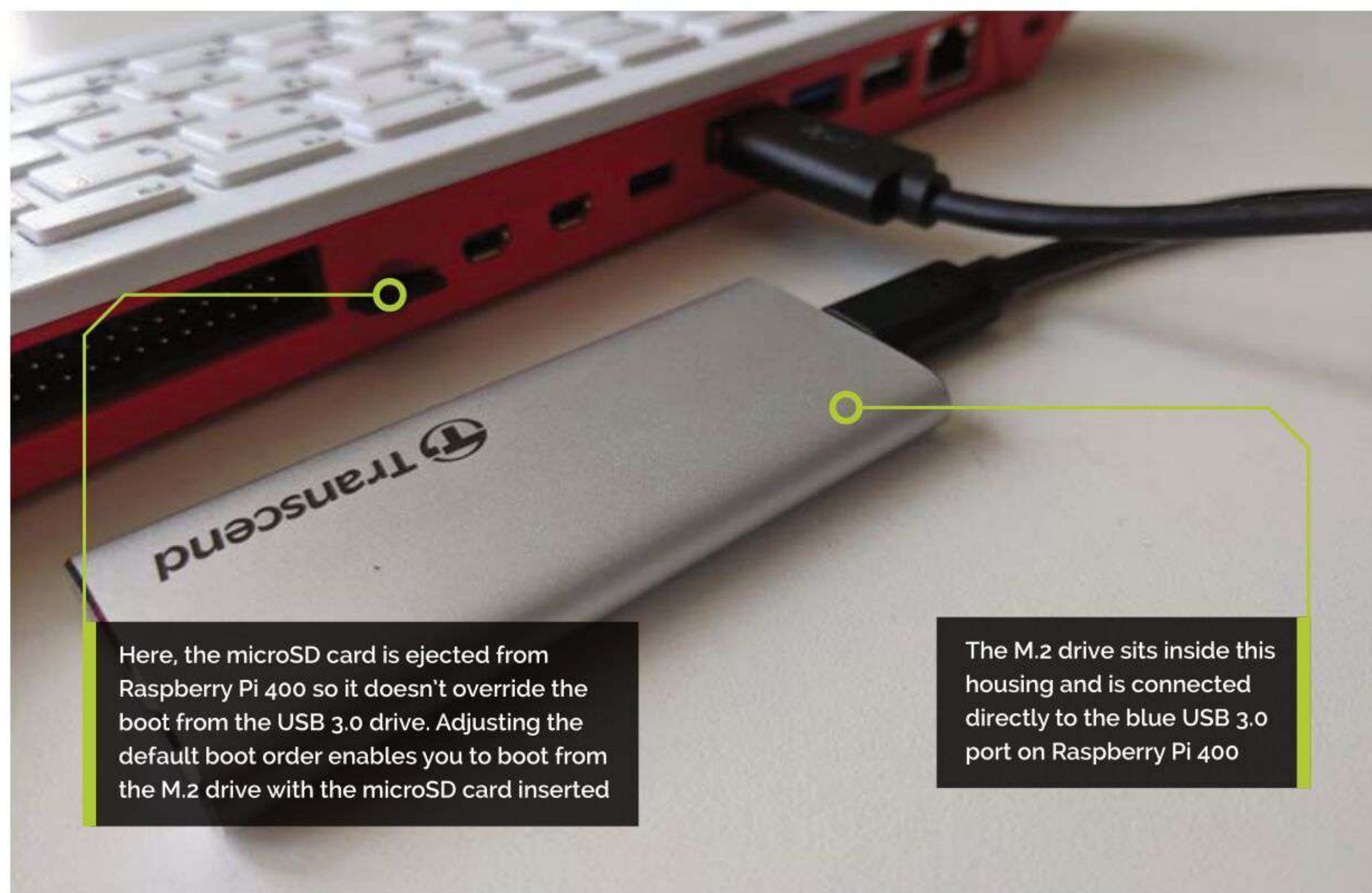
Here, the microSD card is ejected from Raspberry Pi 400 so it doesn't override the boot from the USB 3.0 drive. Adjusting the default boot order enables you to boot from the M.2 drive with the microSD card inserted

The M.2 drive sits inside this housing and is connected directly to the blue USB 3.0 port on Raspberry Pi 400

▲ A Transcend M.2 SSD drive with a SATA III connection (on the left)

enclosure (**magpi.cc/tscm42**). The Transcend 430S was 512GB, a mighty upgrade from the 16GB card included with Raspberry Pi 400. However, you don't need to purchase such a huge drive and the 128GB model will be more than enough for most use cases.

01 Assemble the drive

We start by assembling the M.2 drive enclosure. Our M.2 SATA to USB 3.1 SSD Enclosure Kit (TSCM42S) contains a SATA III to USB board that the M.2 SSD is mounted on. Place the SATA III interface into the socket and gently push the M.2 SSD. Then, a single screw is used to hold the M.2 SSD in place. Once the M.2 SSD drive is affixed to the board, we use the enclosure to contain it. The assembly process will vary depending on which M.2 drive and enclosure you use, but most will follow a similar pattern.

02 Set up the drive

If you want to install a fresh installation of Raspberry Pi OS to the M.2 SSD drive, then use Raspberry Pi Imager (**magpi.cc/imager**) to install the OS directly to the drive. You can do this on any computer, including your Raspberry Pi running from a microSD card. See the 'Using Imager' box (overleaf) and head to Step 4 after installing your fresh installation.

Another option is to boot your Raspberry Pi from the microSD card and clone the current operating system to the M.2 SSD drive. Boot Raspberry Pi 400 from the microSD card and – once Raspberry Pi OS is running – make sure your microSD card is running the latest version of Raspberry Pi OS:

```
sudo apt update
sudo apt full-upgrade
```

03 Copy the drive

Connect the M.2 drive to one of the two blue USB 3.0 connections. Open the Raspberry Pi menu and choose Accessories and SD Card Copier.

Choose the microSD card in Copy From Device; ours is marked 'SC16G (/dev/mmcblk0)'. In Copy To Device, select the M.2 drive. It should be mounted on **/dev/sda** and the only other option available.
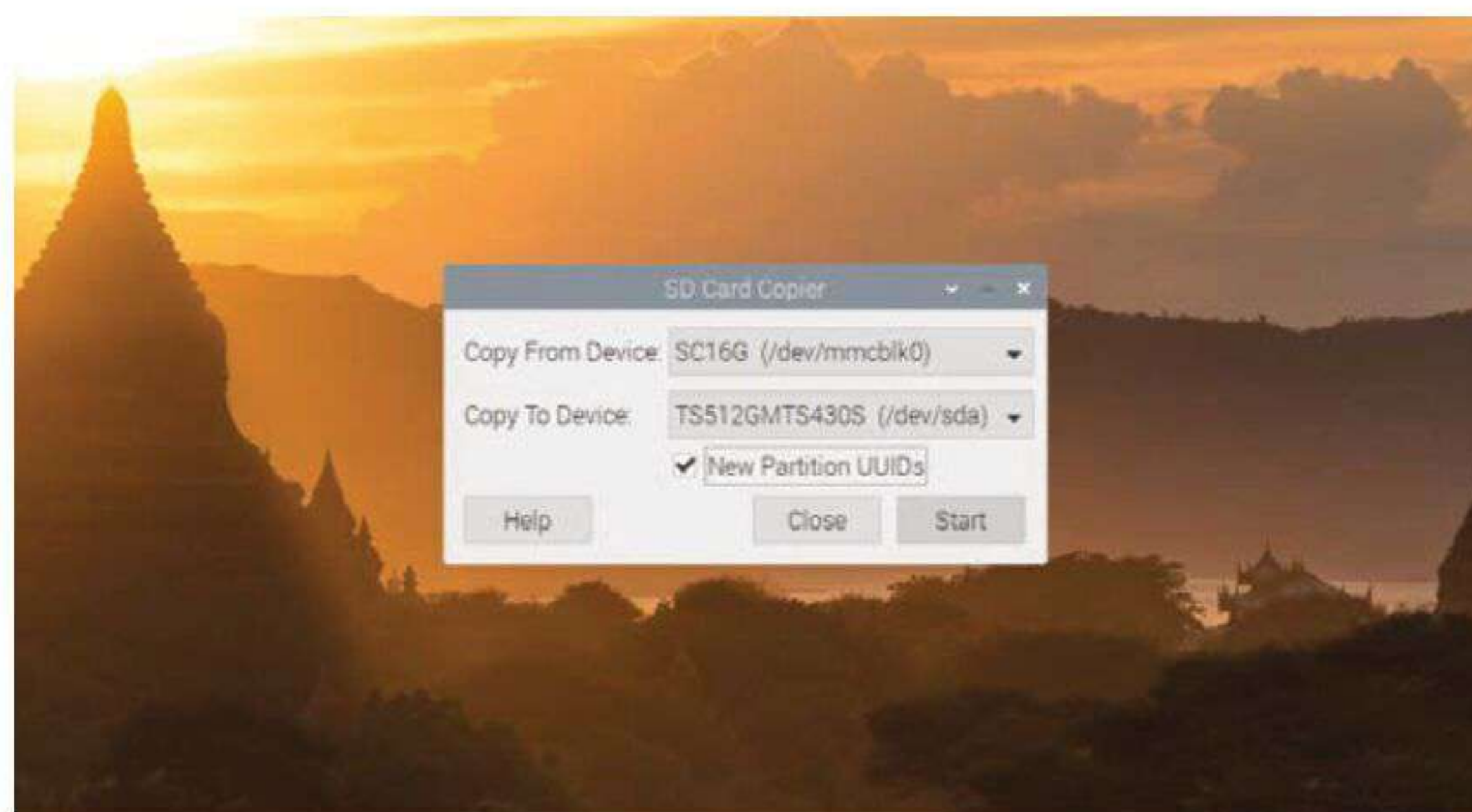
▼ SD Card Copier is used to duplicate the boot image on your microSD card to the M.2 SSD drive
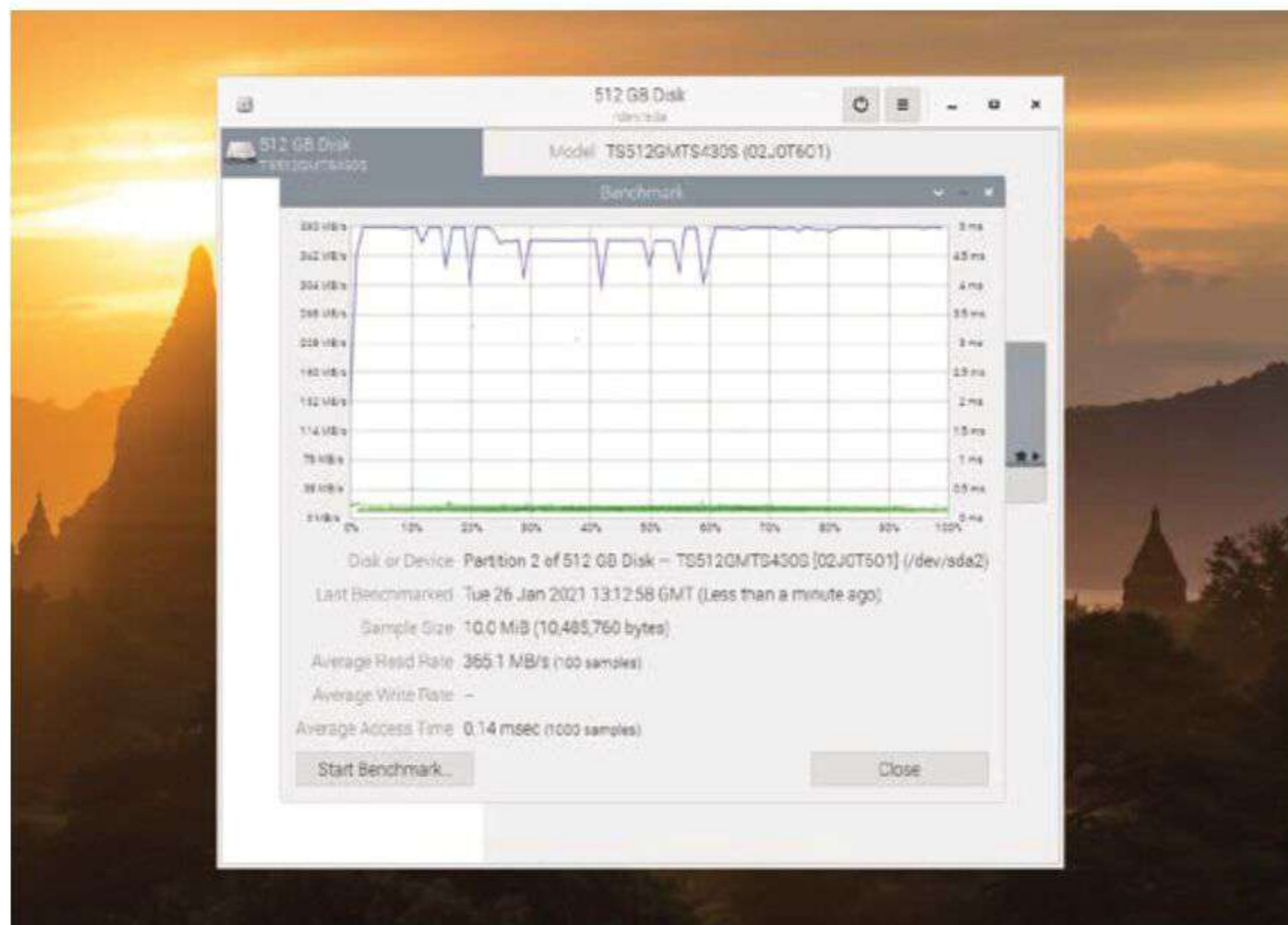

◄ The M.2 SSD adapter translates the SATA III interface on the M.2 SSD drive into a USB-C connection. This is used with a USB-C to USB-A cable to connect the drive to the blue USB 3.0 connection on Raspberry Pi 400

▲ GNOME Disks running a benchmark test that shows the M.2 SSD drive running vastly faster than the microSD card

Using Imager

If you'd prefer to start with a fresh installation of Raspberry Pi OS, it is possible to use Raspberry Pi Imager instead of SD Card Copier. This app will download the latest version of Raspberry Pi OS, format your hard drive, and install the OS (all at the click of a button).

Boot up Raspberry Pi OS using the microSD card and install Raspberry Pi Imager with Terminal:

```
sudo apt update
sudo apt install rpi-imager
```

Choose Raspberry Pi applications menu > Accessories > Imager to open the program. Click on 'Choose OS' and select 'Raspberry Pi OS (32-bit)'. Next, click on 'Choose SD Card' and select your external M.2 drive from the SD Card menu. Click on 'Write' to download the operating system and write a fresh installation to the M.2 drive.

Make sure to tick New Partition UUIDs (this will enable you to mount and access both devices at the same time). Click Start and Yes at the 'erase all content' warning menu to begin the copying process.

04 Boot into M.2

When SD Card Copier has finished duplicating the contents of the microSD card to the M.2 drive, you will be able to use the latter to boot and run your Raspberry Pi 400.

Power off your Raspberry Pi (choose Shutdown > Shutdown from the Raspberry Pi applications menu). Now remove the microSD card from Raspberry Pi as it has boot priority over the external M.2 drive. Press the **FN** and Power (**F10**) keys to power Raspberry Pi 400 back up. It will boot and run from the M.2 drive.

> ❝ You should notice a speed improvement when using the M.2 drive over the microSD card ❞

05 Install GNOME Disks

You should notice a speed improvement when using the M.2 drive over the microSD card. Opening programs and browsing the web will be much faster. To get detailed information about the speed of M.2, you can benchmark the drive with GNOME Disks. Open a Terminal window and install it with:

```
sudo apt update
sudo apt install gnome-disk-utility
```

Open the Raspberry Pi applications menu and choose Accessories > Disks to open GNOME Disks.

06 Speed-test drive

Select the rootfs partition and click the 'Additional partition options' icon (shaped as two cogs); choose Benchmark Partition. Click Start Benchmark and Start Benchmarking to test the

drive. We get an average read rate of 382.4MB/s (much faster than our microSD card).

Insert the microSD card and select it in GNOME Disks to perform a comparative test. We get just 44.9MB/s in comparison.

---

07 **Swap boot order**

You can now use Raspberry Pi 400 with the M.2 drive attached and the microSD card ejected (as it will boot from the M.2 drive). If you insert the microSD card, the EEPROM (electrically erasable programmable read-only memory) in Raspberry Pi 400 will prioritise the microSD card over USB.

Following a recent update to raspi-config, the option to prioritise USB boot over microSD is just a few clicks away. Open Terminal and enter:

```
sudo raspi-config
```

Use the arrow keys to choose Advanced Options and Boot Order, then pick 'B2 USB Boot'. The screen will say 'USB is default boot device'. Press **ENTER** and choose Finish and then Yes to 'Would you like to reboot now?'

When Raspberry Pi reboots, it will start up from the M.2 SDD connected to USB (even if the microSD card is inserted). You can now use your Raspberry Pi 400 with the M.2 SSD drive as the default. 𝕄
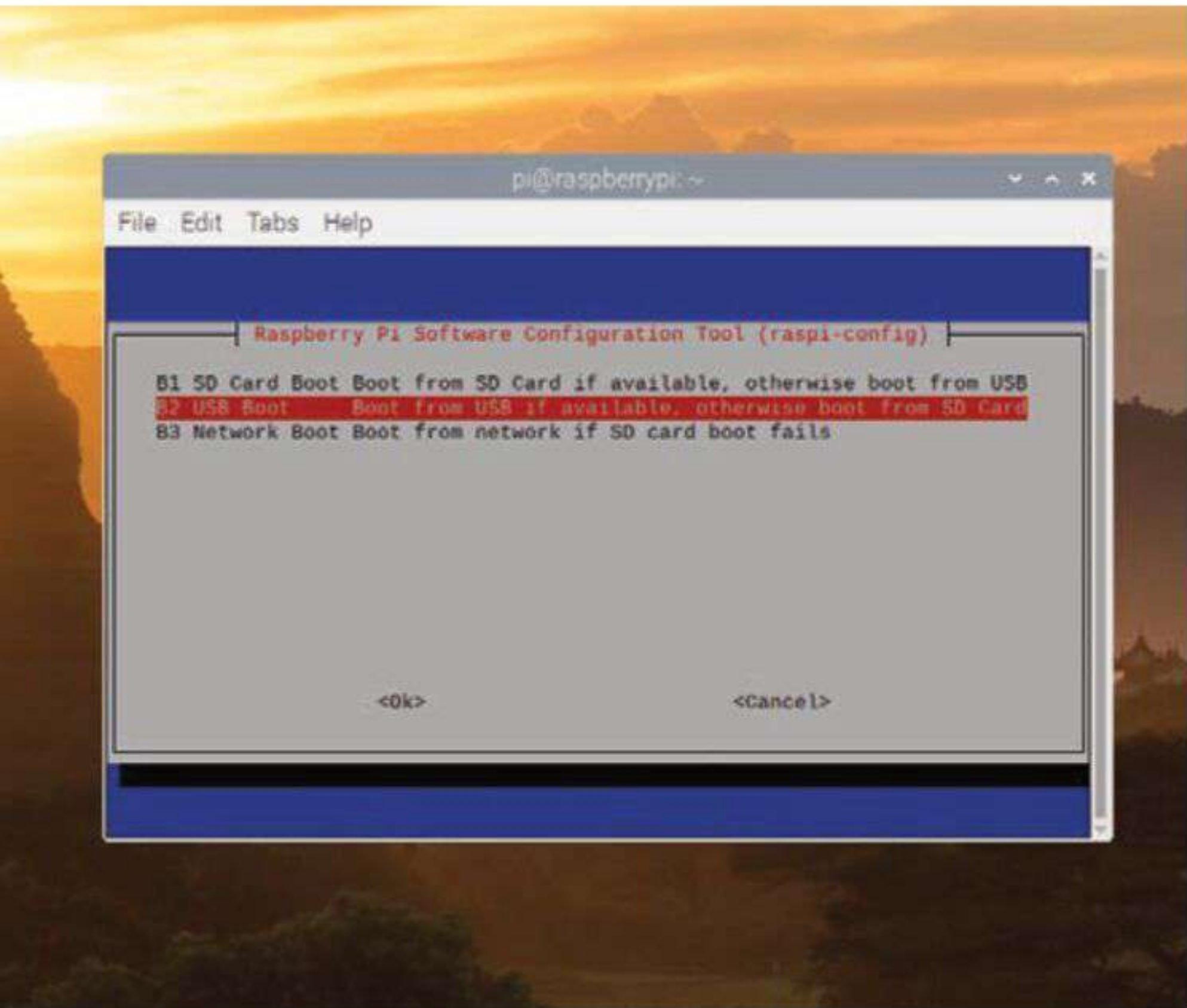
▲ Use raspi-config to set USB as the default boot option

EEPROM and boot order

The option to prioritise USB over microSD can now be found in raspi-config. If you want to see what is happening under the hood (or customise your own boot mode) you'll want to edit the EEPROM configuration.

View the current EEPROM configuration:

```
rpi-eeprom-config
```

This is similar to the **config.txt** file. The last option will be:

```
BOOT_ORDER=0xf41
```

The configuration '0xf41' means try the SD card followed by USB mass storage, then restart. The values after '0x' are read from right to left.

A new boot order
We are looking to change the BOOT_ORDER field to '0xf14'. This will boot from the USB device '4', then the microSD card '1', followed by a restart 'f' if neither is detected.

To edit it and apply the updates to the latest EEPROM release, enter the following in Terminal to open the **boot.conf** file in your text editor (the default is Nano, which we're using here):

```
sudo -E rpi-eeprom-config --edit
```

Change 'BOOT_ORDER=0xf41' to:

```
BOOT_ORDER=0xf14
```

Press **CTRL+O** and then **CTRL+X** to write the file and quit Nano (or save and quit in your preferred text editor). Terminal will report 'EEPROM update pending. Please reboot to apply the update.' Restart your Raspberry Pi to apply the update.

```
sudo reboot
```

BOOT_ORDER fields
The BOOT_ORDER property defines the sequence for the different boot modes. It is read right to left and up to eight digits may be defined.

The default setting is '0xf41'. Read from right to left, this setting is SD CARD, USB-MSD, then RESTART.

Value	Mode	Description
0x1	SD CARD	SD card (or eMMC on Compute Module 4)
0x2	NETWORK	Network boot
0x3	RPIBOOT	RPIBOOT – see magpi.cc/usbboot (since 2020-09-03)
0x4	USB-MSD	USB mass storage boot (since 2020-09-03)
0x5	BCM-USB-MSD	USB 2.0 boot from USB Type-C socket or USB Type-A socket on CM4 IO board (since 2020-12-14)
0xe	STOP	Stop and display error pattern (since 2020-09-03). A power cycle is required to exit this state
0xf	RESTART	Start again with the first boot order field (since 2020-09-03)

Part 02

Cheap Trills for all

Using the ring Trill sensor, see how you can make a
wind-up music box with replaceable music cylinders

MAKER

Mike Cook

Veteran magazine
author from the old
days, writer of the
Body Build series,
plus co-author of
*Raspberry Pi for
Dummies, Raspberry
Pi Projects*, and
*Raspberry Pi Projects
for Dummies.*

magpi.cc/mikecook

Last month we looked at how to interact with
the Trill sensors using Python. Now we'll
show you how to use these sensors in three
different projects: specialised mouse control, a
jukebox player, and your very own virtual wind-up
music box.

01 Smart mouse control
One of the most useful things you could
add to a piece of established software is smart
mouse control. For example, take an application
like the 3D modelling application Blender. It is
quite a dense interface requiring you to remember
multiple key combinations to get into different
modes. In our opinion, one of the most confusing
is the number of different types of dragging you
can do (see **Figure 1**). How about if we had some
touch sensors, so that each touch sensor controlled
one type of drag or zoom?

Action	Result	Figure 1
Left-click	Select	
Shift+left-click	Add to selection	
Ctrl+left-click (edit mode)	Remove from selection	
Left-click+drag	Box selection	
Alt+left-click (edit mode)	Edge/Face loop select	
Middle-click+drag	Rotate view	
Shift+middle-click+drag	Pan view	
Ctrl+middle-click+drag	Zoom view	
Right-click	Context menu	

▲ **Figure 1** Actions in Blender

02 PyAutoGUI
The difficulty is how do we get a Python
program to control the mouse? Enter PyAutoGUI,
an extension that automates graphical user
interface control. You can obtain it easily by going
to Add/Remove Software in Raspberry Pi OS's
Preferences menu and searching for it. Scroll down
to the name, tick the box next to the name, and
click on the Apply button; after a short time, you'll
have it. No need to even reboot your machine.

03 Direct mapping
The simplest way of using a Trill sensor is
by mapping the output of one directly to what you
want to control. The positional output for a sensor
is a floating point number that goes from 0 to 1.0,
so mapping is simple. For mouse movements this
means multiplying the sensor's X and Y outputs
by the screen width and height, to get the mouse
position to move to, plus detecting a second touch
on the sensor to trigger a click. We show you how
in the **trill_simple_mouse.py** program (download
all the code from **magpi.cc/pibakery**).

04 Incremental mapping
You can get a remarkable degree of control
by rolling your finger rather than dragging it, but
using the direct mapping can be a bit coarse for
large screens. On laptops, the touchpad control
often covers only part of the screen. To do this, a
program has to add any new movement from the
Trill sensor to the current mouse position. We show
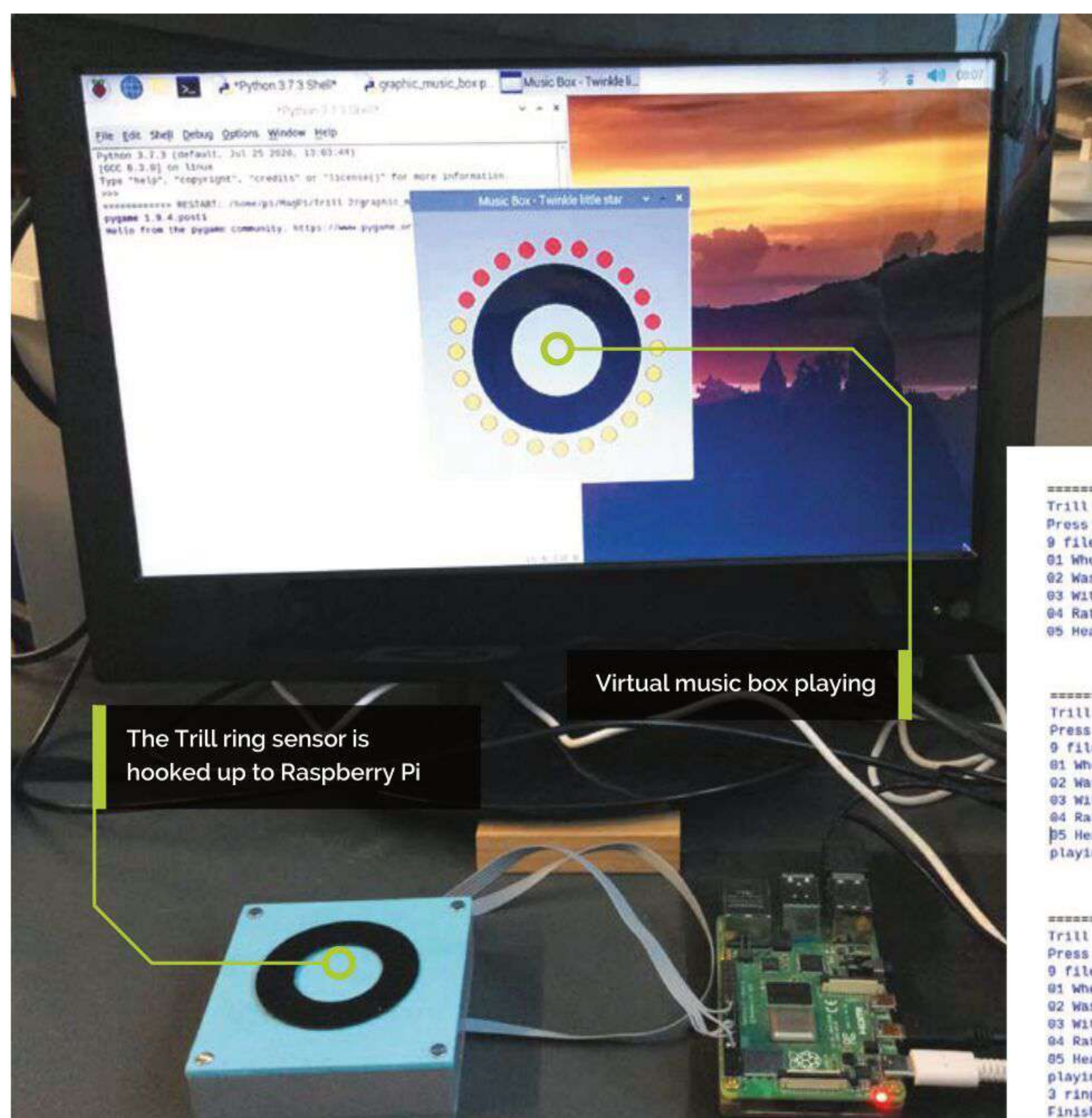you how to do this in the **trill_mouse.py** code.

The Trill ring sensor is hooked up to Raspberry Pi

Virtual music box playing

Figure 2

```
================= RESTART: /home/pi/MagPi/Trill 2/jukebox.py =================
Trill Ring Jukebox
Press Ctrl-C to stop.
9 files found
01 When You Finish Me.wav
02 Wasteland.wav
03 Witching Stone.wav
04 Rats.wav
05 Heaven And Hell.wav


================= RESTART: /home/pi/MagPi/Trill 2/jukebox.py =================
Trill Ring Jukebox
Press Ctrl-C to stop.
9 files found
01 When You Finish Me.wav
02 Wasteland.wav
03 Witching Stone.wav
04 Rats.wav
05 Heaven And Hell.wav
playing  05 Heaven And Hell.wav


================= RESTART: /home/pi/MagPi/Trill 2/jukebox.py =================
Trill Ring Jukebox
Press Ctrl-C to stop.
9 files found
01 When You Finish Me.wav
02 Wasteland.wav
03 Witching Stone.wav
04 Rats.wav
05 Heaven And Hell.wav
playing  05 Heaven And Hell.wav
3 ring sensor touches - stopping
Finished playing
```

▲ **Figure 2** Using the touch-wheel jukebox

Finally, by adding a ring sensor to the square sensor, you can emulate a scrolling centre button mouse – see the code **trill_simple_mouse_scroll.py** for how to do this.

05 Simple jukebox

We can use the ring sensor alone, to control the playing of music files, stored in the top-level **Music** directory with the **simple_jukbox.py** code. Here, the **Music** directory is scanned to compile a list of all files it contains. Then, by using the Trill ring sensor, you can show them one at a time in the console. Adding a second touch to the sensor will then play that track using the built-in OMXPlayer. The track will then be played to the end. Stopping a track isn't quite so simple.

06 More control

In order to have more control over OMXPlayer, you have to use omxplayer-wrapper – installed with this Terminal command:

```
pip3 install omxplayer-wrapper
```

This allows you to send control messages to the player while it is playing a track. So we used that feature to stop a track while one was playing, by giving three touches on the ring. This turned out to be quite tricky because once a track has stopped, calls to read the status of the track crash the code. The code **jukebox.py** shows how we did this, and **Figure 2** shows how it looked.

> **❝ Run your finger round the ring to emulate winding up the music box ❞**

07 A music box

All the other projects here do not use a graphical user interface, but our final one does. The code is called **graphic_music_box.py**. The screen shows a Trill ring sensor surrounded by a halo of 24 coloured circles representing LEDs (**Figure 3**). Operation is simple: run your finger round the ring to emulate winding up the music box. As you wind, the colours change for each full turn of the key. When the music box starts off, all the virtual LEDs are off – i.e. the same colour as the background; the first full turn of the key turns them red.

You'll Need

> Trill ring sensors
> **magpi.cc/trillring**

> Trill square sensors
> **magpi.cc/trillsquare**

Figure 3 — Music Box - Twinkle little star
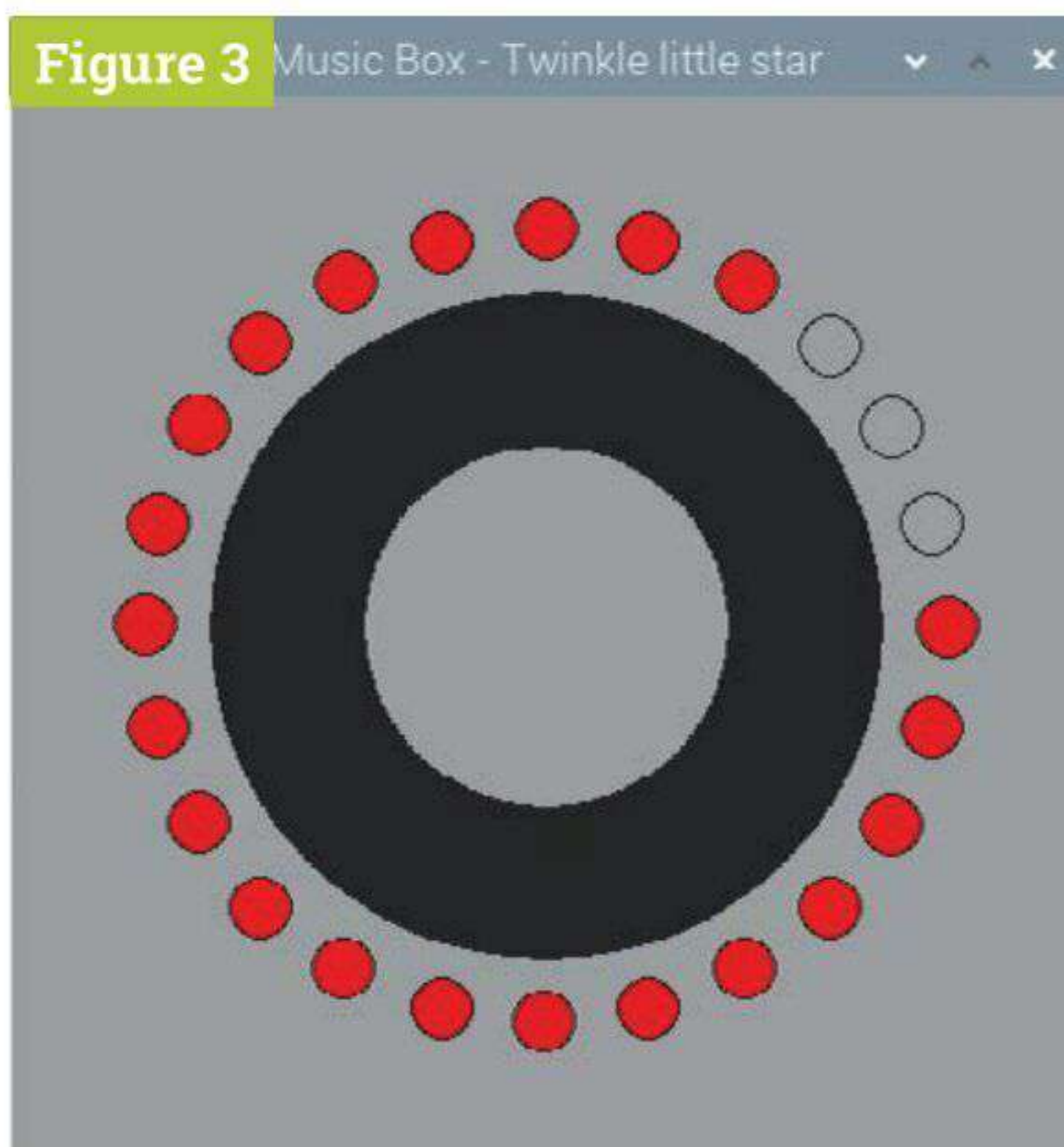
08 Winding it up

When all the LEDs are white, the box is fully wound at five turns, and no further winding can occur. To start playing, simply remove your finger from the ring. With less than twelve LEDs' worth of wind left, the tune becomes increasingly slower, and then stops. A real music box has the tune encoded with a metal cylinder covered in spikes; in the higher-end boxes, these cylinders can be changed. With our virtual music box, this can be achieved with a double touch on the ring. The name of the new tune will be displayed as the window name.

Top Tip 👍

Making the Trill ring box

The drawings for the ring box can be found on our GitHub page, along with all software and data files.

09 The sounds

To create the sounds of the music box, we recorded notes from a MIDI sound module one at a time using Audacity, the free sound recorder application. When downloading it with the Add/Remove Software tool, make sure to install both the application and the data files. Then, we saved each note separately as a WAV file, giving it the name of the MIDI note number it represents. We recorded notes in the range of 35 to 84, or B in octave −3 to C in octave 2. These were placed into a folder called **music_box_sounds**.

10 The cylinders

The cylinders are files stored in the surprisingly named folder **cylinders**, with the extension .cyl. In fact, these are files with a simple format that you can generate yourself: they consist of a text file, with a top line that contains the name of the song, a number containing the tempo, and the number 1, all separated by commas. The rest of the file comprises lines of three numbers, a time stamp, a MIDI note number, and a velocity or volume. This last number is used to tell if it is a note on or off message: zero for off, non-zero to sound.

11 Creating your own cylinders

While we have six sample cylinders on our GitHub page, it is fun to create your own. These are formed by taking a MIDI file and stripping it down to just the information required. To do this, we need to use a little application called 'midicsv', obtainable via Add/Remove Software. It can only be used from the Terminal window by navigating to the folder where the .mid file is stored and typing:

```
midicsv fileName.mid saveName.csv
```

Then we can open up the CSV file in a spreadsheet application and delete the information we don't need.

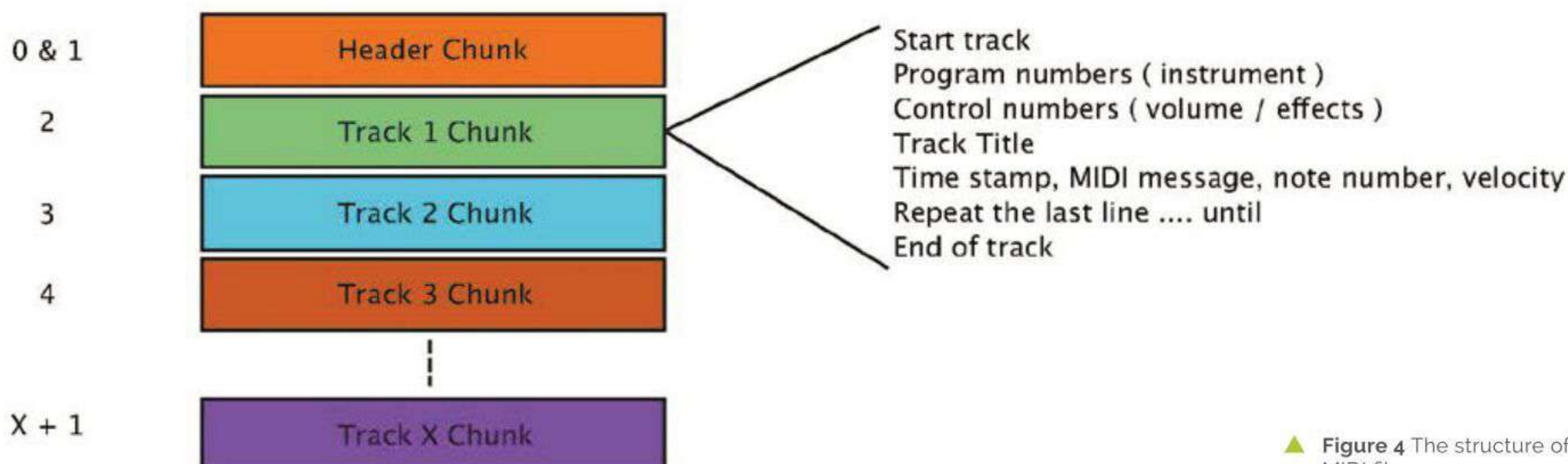**Figure 4**

Chunk Number	Standard MIDI File format
0 & 1	Header Chunk
2	Track 1 Chunk
3	Track 2 Chunk
4	Track 3 Chunk
X + 1	Track X Chunk

Start track
Program numbers (instrument)
Control numbers (volume / effects)
Track Title
Time stamp, MIDI message, note number, velocity
Repeat the last line .... until
End of track

▲ **Figure 4** The structure of a MIDI file

Figure 5

Remove columns

greensleevees

0	0	Header		1		3	256
1	0	Start_track					
1	0	Time_signature		1		3	12	8
1	0	Key_signature		0	"major"		
1	0	Tempo	727271				
1	0	Title_t	"Greensleeves"				
1	0	Text_t	"Traditional"				
1	12288	Tempo	1333331				
1	12288	Tempo	727271				
1	12288	Time_signature		6		3	36	8
1	23744	Tempo	888887				
1	23936	Time_signature		5		3	12	8
1	24320	Tempo	2000000				
1	24321	End_track					
2	0	Start_track					
2	0	Program_c		0	24		
2	0	Control_c		0	121	0	
2	0	Title_t	"Guitar"				
2	0	Note_on_c		0	57	66	
2	128	Note_off_c		0	57	0	
2	128	Note_on_c		0	60	78	
2	128	Note_on_c		0	45	66	
2	384	Note_off_c		0	60	0	
2	384	Note_on_c		0	62	81	

Remove
Rows

To end of file

2	24320	Note_on_c		0	57	76
2	24320	Note_on_c		0	45	77
2	24576	Note_off_c		0	57	0
2	24576	Note_off_c		0	45	0
2	24577	End_track				
3	0	Start_track				
3	0	Program_c		1	24	
3	0	Control_c		1	121	0
3	0	Control_c		1	64	0
3	0	Control_c		1	91	48
3	0	Control_c		1	10	64
3	0	Control_c		1	7	100
3	0	Title_t	"Guitar"			
3	0	Note_on_c		1	57	68
3	128	Note_off_c		1	57	0
3	128	Note_on_c		1	60	79
3	128	Note_on_c		1	45	67
3	384	Note_off_c		1	60	0
3	384	Note_on_c		1	62	75

12 Hacking a MIDI file

A MIDI file consists of a header, along with separate information for each instrument or track – the format is shown in **Figure 4**. As our music box has only one voice, we are looking for MIDI files that contain only one track, or most of the tune in one track. **Figure 5** shows part a MIDI file opened up in a spreadsheet application, annotated to show you what information to remove. When you have removed these bits, save what remains as a CSV file. Open this in a text editor, add the first line, and save it as a .cyl file.

13 Tempo

Note that when using a two-dimensional sensor, like the square and the hex, it should be considered as a single touch sensor only. While it can detect multiple touches, it does so by working out the number of horizontal and vertical touches detected; there can be a different number of these, and it can't pair them up. However, more advanced tracking algorithms could be devised for things like gesture recognition.

So, now you have enough information to start using the Trill touch sensors. Next month we will see how to mount these sensors into a project, and look at some of the things you can do with them.

> ## We particularly like the results when playing The Blue Danube

14 Finally

With such a wide range of notes, some of the lower notes sound much more clunky than others. We particularly like the results when playing *The Blue Danube*: it sounds like it is being played by a bag of spanners. A good source of free, simple MIDI files, is **8notes.com** – we found it best to choose arrangements for single monotone instruments like wind or brass.

Using a Trill ring and a 24 LED ring which fits nicely round it, along with a battery, amplifier, and speaker, you could implement this as a standalone project using a Raspberry Pi Pico microcontroller. M

▲ Figure 5 Data to remove from spreadsheet view of MIDI file

Top Tip

Cylinder note range

Notes outside the range of 35 to 84 are flagged while loading a cylinder file. Lower notes result in the lowest note being played, but higher notes will crash the program.

EASY
Raspberry Pi Pico
PROJECTS

Don't be intimidated by your new Raspberry Pi Pico microcontroller.
Here are some great projects being built with Pico. By Lucy Hattersley

Raspberry Pi Pico is a brand new microcontroller from Raspberry Pi, shaking up the world of computing. We covered Pico in-depth last issue (**magpi.cc/102**) and we're as keen as you are to get to grips with Pico and start making stuff. Microcontrollers are a new piece of technology for Raspberry Pi and *The MagPi* magazine, and coding for them is slightly different from a Raspberry Pi running Raspberry Pi OS.

Fortunately, there are a lot of people out there making Pico life a lot easier. A wide range of Pico products and add-ons are being created. These can be used to add screens, lights, buttons, and audio to Raspberry Pi Pico. Along with code examples and the excellent documentation, these make Pico burst with potential.

There are trailblazers leading the way, with projects and examples of how to make the most of Pico, with its GPIO pins and interesting PIO (Programmable Input/Output) technology.

In this feature, we're going to look at some of the things people are making with Pico, and some easy ideas for projects to build.

❝ There are a lot of people out there making Pico life a lot easier ❞

TIP

SOLDERED HEADERS
To test out Raspberry Pi Pico projects, and use most of the kit mentioned in this feature, you'll need to solder header pins onto your Raspberry Pi Pico. See *The MagPi* #102 (**magpi.cc/102**) for a soldering guide. Alternatively, Pimoroni or SB Components both now sell Raspberry Pi Pico devices with pre-soldered headers.
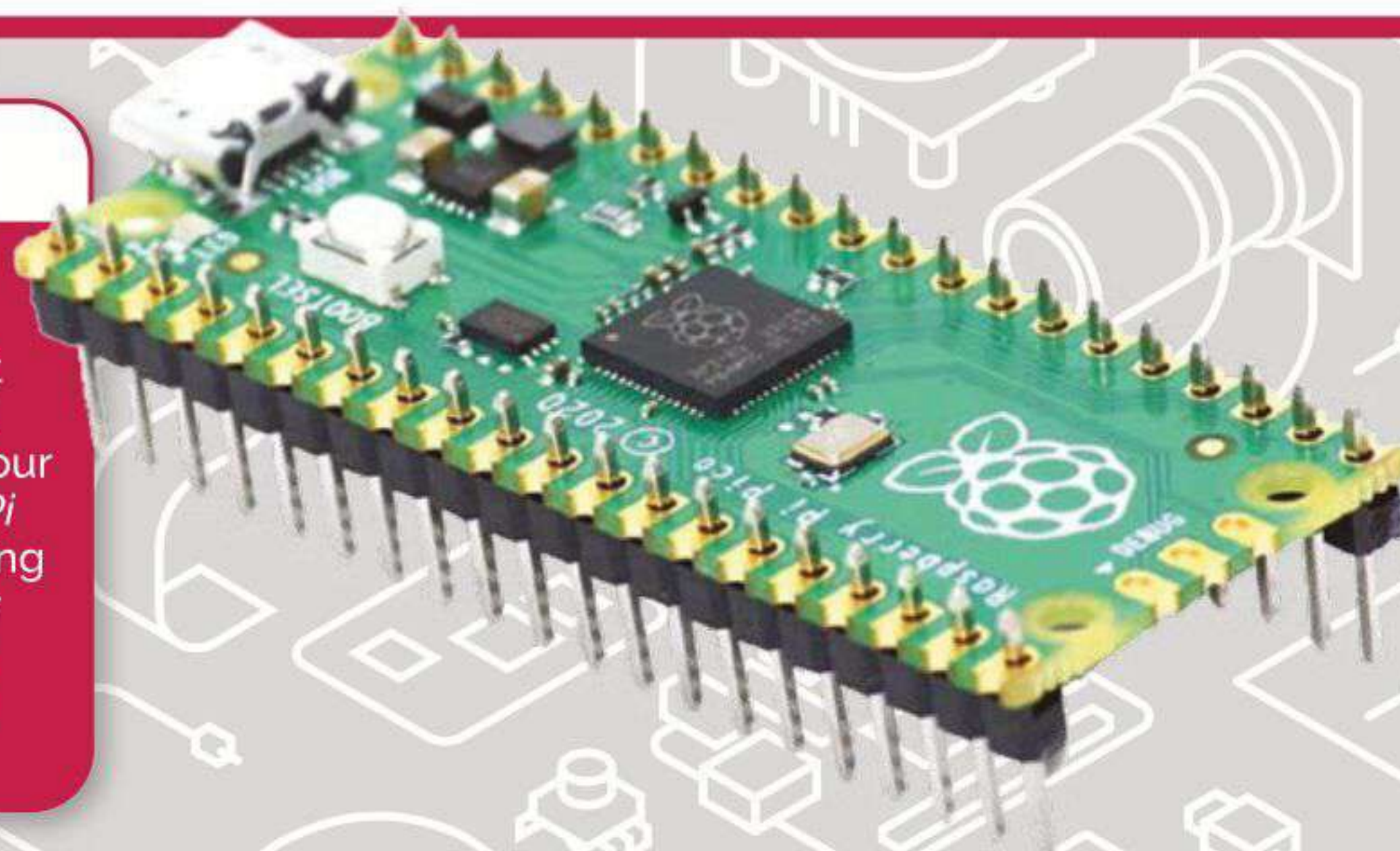
The functionality of Pico can be expanded directly with add-ons such as the Pico Display Pack, which adds a 1.14-inch IPS LCD display and four buttons

The breadboard is used for quickly adding and removing components without soldering. Some breadboards are being custom-built for using with Raspberry Pi Pico. The Pico Breadboard Kit has GPIO pins for the soldered Pico along with buttons, LED lights, and a buzzer
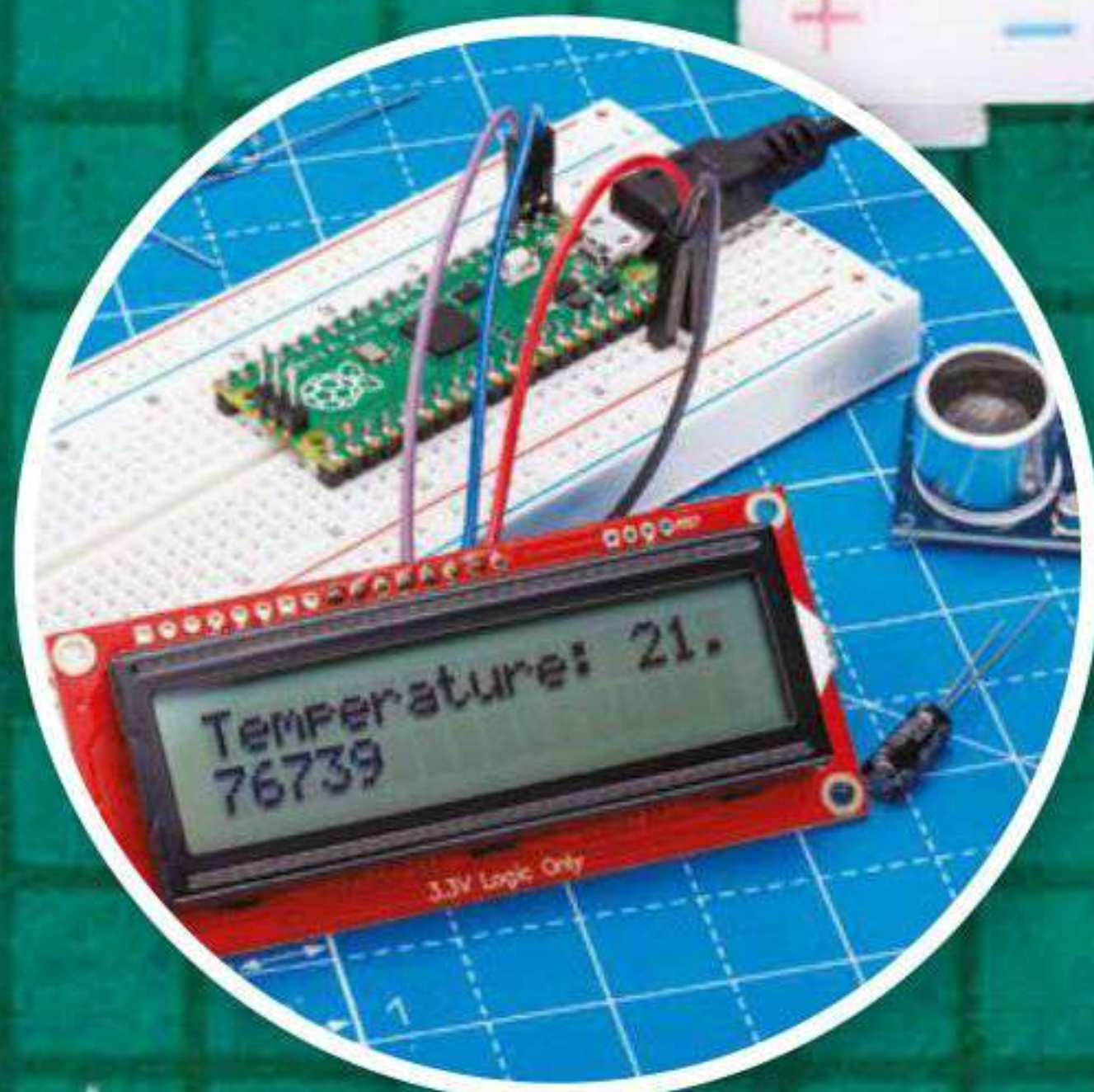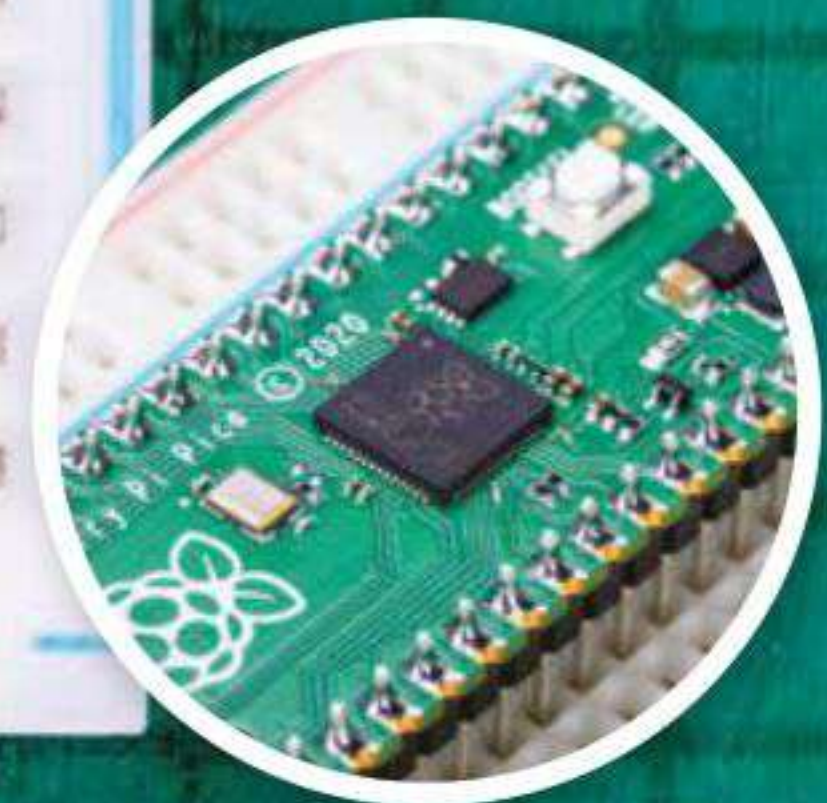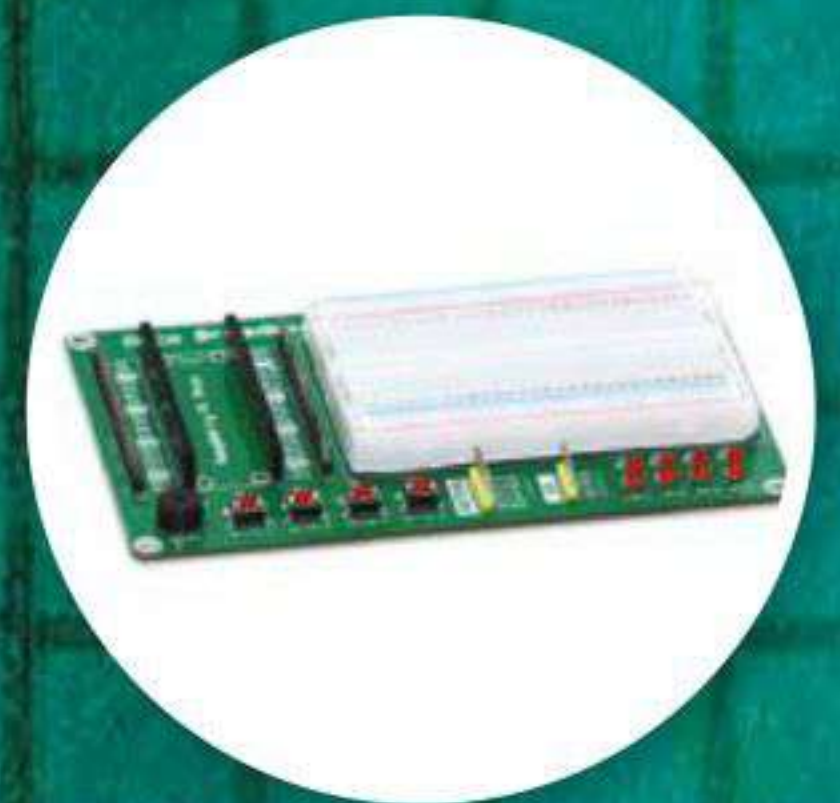
Electronic components, like the SparkFun SerLCD display and sensors pictured can be tested out in the breadboard, and then wired up directly to Pico

Soldered pins enable you to attach Raspberry Pi Pico to a breadboard for experimentation, and also connect new hardware add-ons designed for Pico. Pimoroni and SB Components are selling Pico boards with pre-soldered headers if you want to skip the process

GET VISUAL WITH YOUR PROJECTS

Bring Pico to life with these visual and sound-based projects

Raspberry Pi computers have HDMI output and 3.5 mm minijack sockets to hook up speakers and headphones. This makes it easy to get visual and aural feedback from your programs.

Raspberry Pi Pico is more lightweight. It is possible to create a DVI output using PIO on Pico, but most newcomers start by blinking the on-board LED and send the classic Hello World output to a terminal on another computer (**magpi.cc/hellopico**). Beyond that, you'll want to start adding electronic kit to get the most from Pico.

Pixel strips

If you want to create a light show, then NeoPixels (also known as WS2812B LEDs) are cheap and widely available strips (**magpi.cc/neopixels**). Each strip contains an array of red, green, and blue LED lights designed to be controlled by a microcontroller. As you'd expect, Pico is perfect for controlling NeoPixel light displays.

HackSpace magazine's Ben Everard has written an article for Raspberry Pi on using NeoPixels with Pico (**magpi.cc/neopixelpico**) and you can take a look at the code on GitHub (**magpi.cc/picolightsgit**).

Visual add-ons

If you want something custom-built for Pico, a range of light displays, small screens, and sound boards are already on sale. Few folks have stepped up to the Pico platform like Pimoroni (**magpi.cc/pimoronipico**).

Pimoroni has produced two LED add-ons: the Unicorn Pack is a 16×7 grid of colour LEDs (**magpi.cc/picounicorn**) and the Scroll Pack (**magpi.cc/picoscroll**) is a 17×7 grid of

▼ The Pico Unicorn Pack can be used to create dazzling projects

white LEDs. The Unicorn Pack can be used to create a rainbow display for your window (**magpi.cc/rainbow**), as well as simple animations. The Scroll Pack can be used to create scrolling text messages (hence the name), and it's also great for displaying graph data and for creating a status light.

In this issue we have our first project that makes use of a Pimoroni Pico Pack. On page 40, learn to use a Unicorn Pack to build a Pomodoro Timer (a productivity tool).

Mood lighting is another great use for LED light displays; the white LEDs of the Scroll Pack or Unicorn Pack can be used to light up areas around the house, or to throw a spot of colour into a specific area, like a display case.

If you want something larger, the PicoPythonHub75 project (**magpi.cc/picohub75**) bounces the word 'Pico' up and down on a Hub75 32×32 RGB LED panel. Hub75 panels are an affordable way of adding lots of colourful lights to a build, and Pico's PIO output enables super-fast animations.

LED displays turn up in a range of Raspberry Pi projects, partly because a small 8×8 RGB LED display was included with the Sense HAT for Raspberry Pi. The Sense HAT projects page (**magpi.cc/sensehatprojects**) is packed with ideas that could easily be adapted to a Unicorn Pack or Hub75 LED display. Everything from a countdown timer to a magic 8-ball, advent calendar, and weather logger could be adapted from Raspberry Pi to Raspberry Pi Pico.

Adding displays

Pimoroni's impressive Pico Display Pack (**magpi.cc/picodisplaypack**) adds a small 1.4-inch IPS LCD display to the board, with four buttons and a single RGB LED. This can be used to make a small photo display, or short animations, and give feedback on data. You can create a digital thermometer using Pico's internal temperature sensor (see chapter 8 of the *Get Started with MicroPython on Raspberry Pi Pico* book, **magpi.cc/picobook**).

Once you've got a screen on Pico, you can connect sensors and display the output, opening
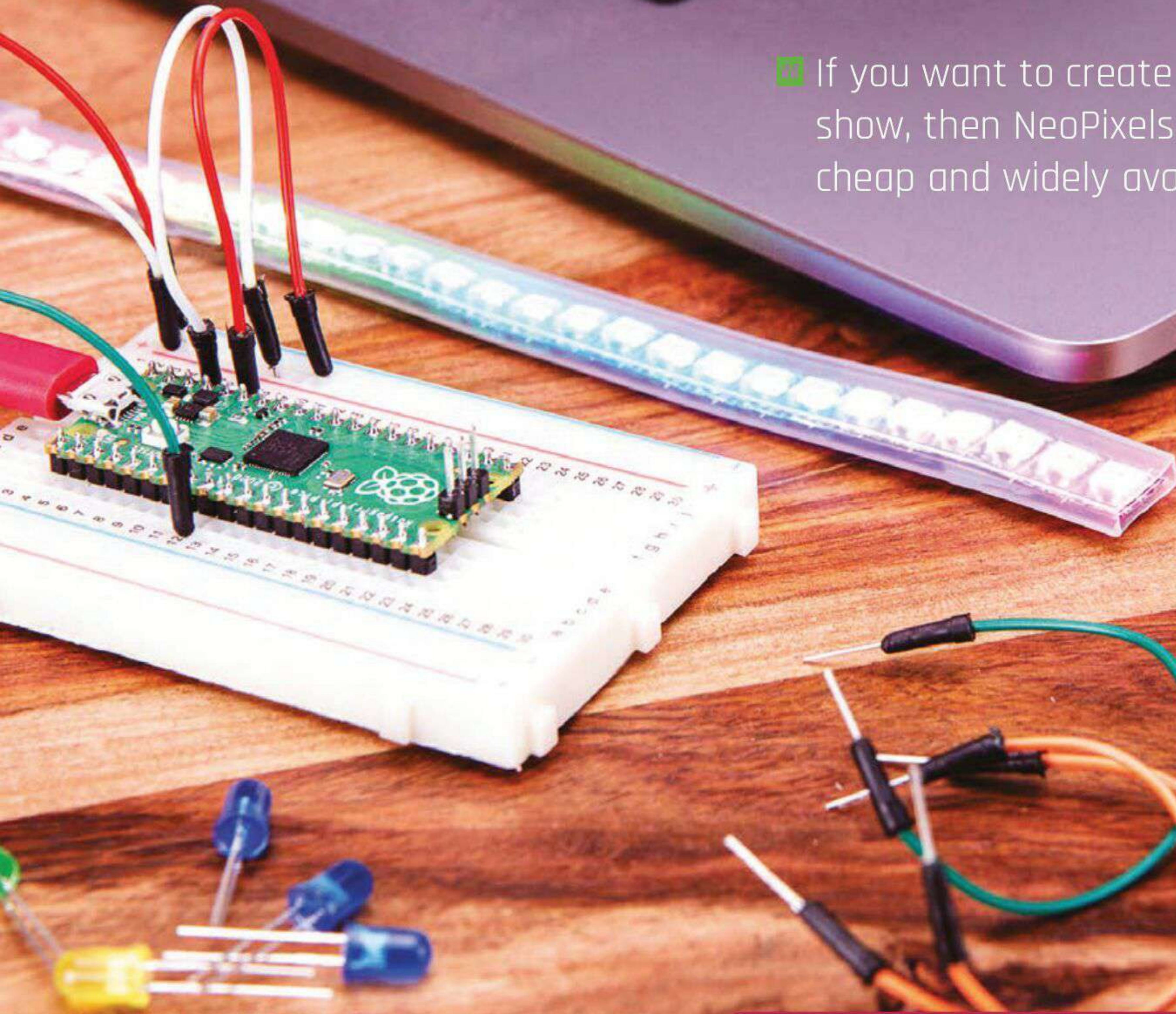
▲ Create mood lighting controlled by code with Raspberry Pi Pico and NeoPixels
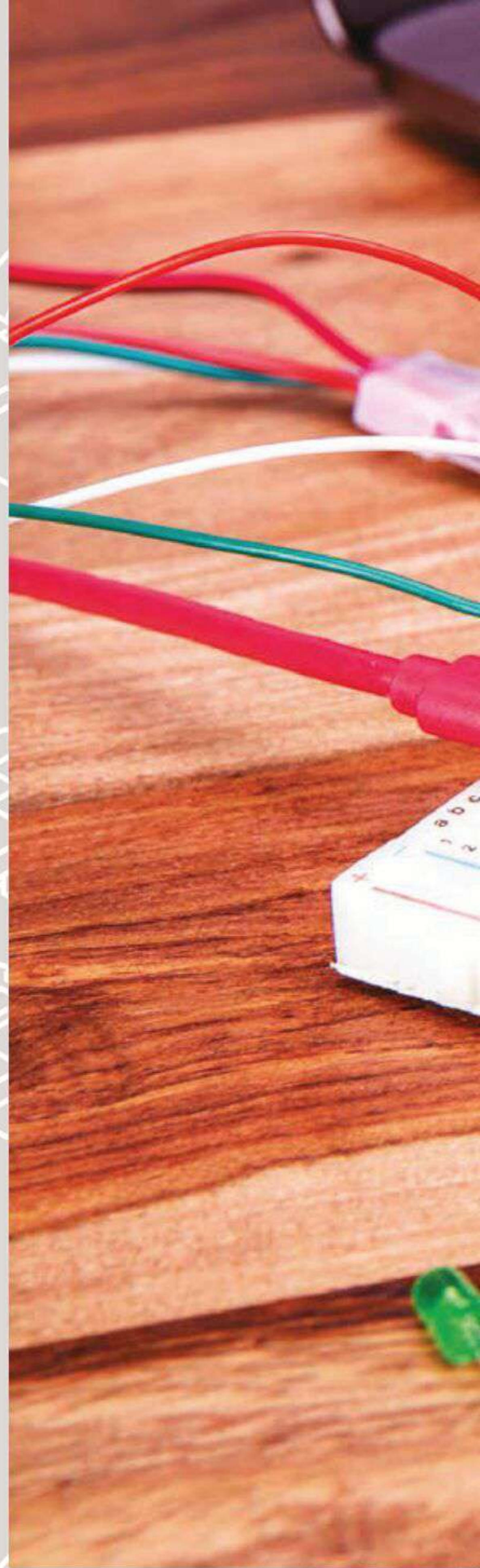
If you want to create a light show, then NeoPixels are cheap and widely available

## 3D-PRINTED CASE FOR RASPBERRY PI

Adafruit has created a Lego-compatible mount for Raspberry Pi Pico that adds studs to our favourite microcontroller board. This mount fits standard bricks or base plates, enabling you to affix Pico into a Lego project (**magpi.cc/picolegocase**).

The mount features built-in standoffs so you can snap fit the Raspberry Pi Pico on top. Taking the project further, The Ruiz Bros have made a Pico Enclosure with lights, buttons, a slide switch, and a small LiPo battery. You can download the STL files and see instructions on Adafruit's website (**magpi.cc/pico3dprintedcase**).

up a range of IoT (Internet of Things) and household projects. We look at some of these in the electronics projects later, but if you're going to attach sensors with Raspberry Pi Pico, a screen to display feedback is a great add-on.

Many of the gaming tutorials later in this feature make good use of the Pimoroni Display Pack. If you'd prefer to wire up a small display directly, then it's relatively easy to connect a SparkFun SerLCD (**magpi.cc/serlcd**) to Raspberry Pi Pico. You can wire it up to the Inter-Integrated Circuit (I2C) or Serial Peripheral Interface (SPI) pins on Pico, and there are control methods implemented in MicroPython. These small displays can provide feedback from other electronic devices. A good starter project is to display the temperature of Pico's built-in temperature sensor. Take a look at chapter 10 of *Get Started with MicroPython on Raspberry Pi Pico* (**magpi.cc/picobook**). M
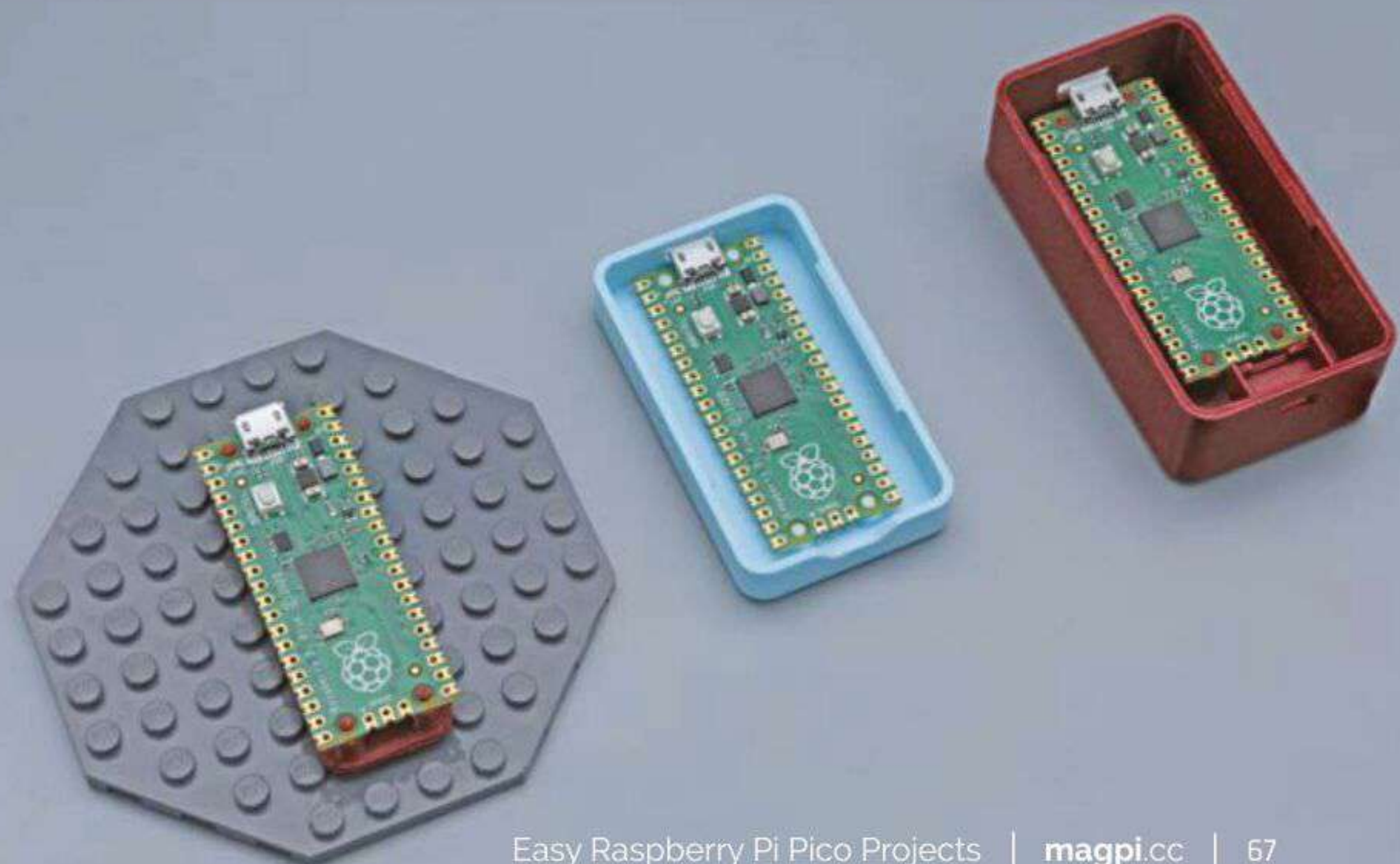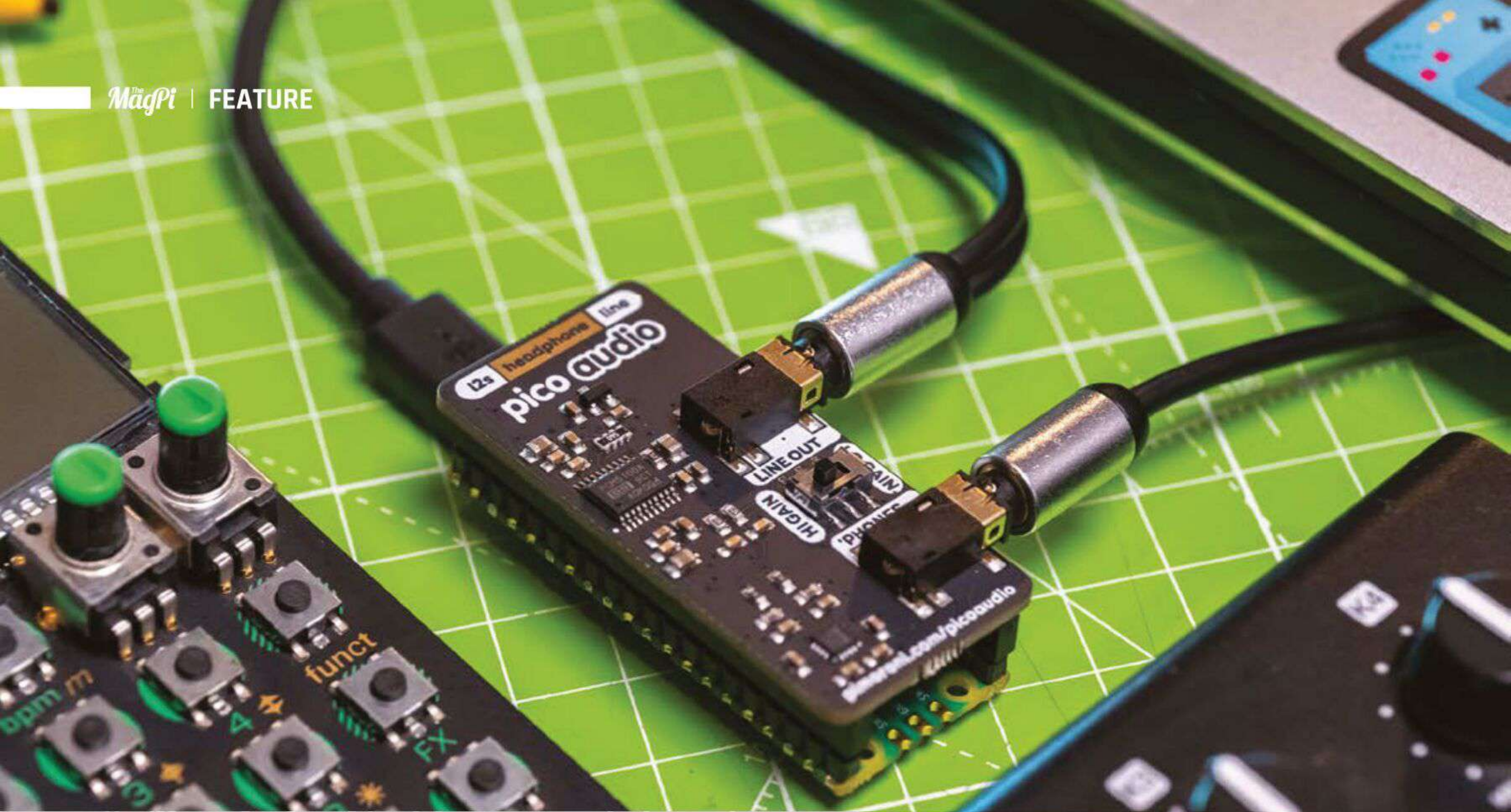
AUDIO PROJECTS

Add some sounds to your projects

When it comes to audio output, Raspberry Pi Pico is silent. There is no built-in speaker or 3.5mm minijack socket. All is very much not lost, however.

Ben Everard, from *The MagPi*'s sister publication, HackSpace magazine, has produced an excellent PIO Buzz tutorial (**magpi.cc/piobuzz**) that explains how to wire up a speaker or headphones to PIO (Programmable Input/Output). Pico's PIO allows developers to define new hardware features in software – expanding its capabilities beyond any fixed-function device.

Ben says it's "a simple PIO program that outputs a tone based on a number going into it. I've not yet done the maths to work out what number means what note, but it seems to give a nice range over audible tones. A bit buzzy at lower frequencies, but sounds nicer higher up."

You might not be able to play music tracks through it, but PIO Buzz is a good way to add audio feedback to your projects. The tones can be used to create alerts, notifications, and provide user interface feedback in projects.

Audio playback

If you're looking for audio file playback, then the Pico Audio Pack (**magpi.cc/picoaudiopack**) will be of interest. It adds a PCM5100A DAC (digital-to-analogue converter) to output up to 32-bit, 384kHz stereo audio through a 3.5mm line-out connector. It can also pump out amplified stereo from a 3.5mm headphone jack.

With the Pico Audio Pack, you can turn Raspberry Pi into a lo-fi synthesizer and generate interesting noises in code. If you're creating synth projects, you might also want to look at the Pico RGB Keypad Base (**magpi.cc/picorgbkeyboard**). This is a 4×4 row of illuminated keys that can be used to create a USB keyboard device. These are popular amongst video streamers and DJs. Use it to

> ❝ Create a disco dance floor with your fingers ❞

"create a disco dance floor with your fingers," says Pimoroni, "or a Simon Says-style game with which to taunt your friends."

One thing to note with the Pico Audio Pack is that it works with Pico's C/C++ SDK (with MicroPython support planned). So right now you'll need to invest some time in the more challenging code base. Still, the possibilities for creating a Pico-based digital music player, Simple Simon game, DJ system, or DAC for your home sound system are worth the investment. **M**

▲ Pico Audio Pack

▶ The Pico RGB Keypad can be combined with the Audio Pack to create devices that respond to finger taps. Perfect for DJs and video streamers

THREE EASY PICO PROJECTS ON YOUTUBE

Print 'N Play is a YouTube channel dedicated to building and making. This video covers setting up Raspberry Pi Pico and moves on to a range of easy electronics projects. The code for each project is available on GitHub.
magpi.cc/3easyprojects
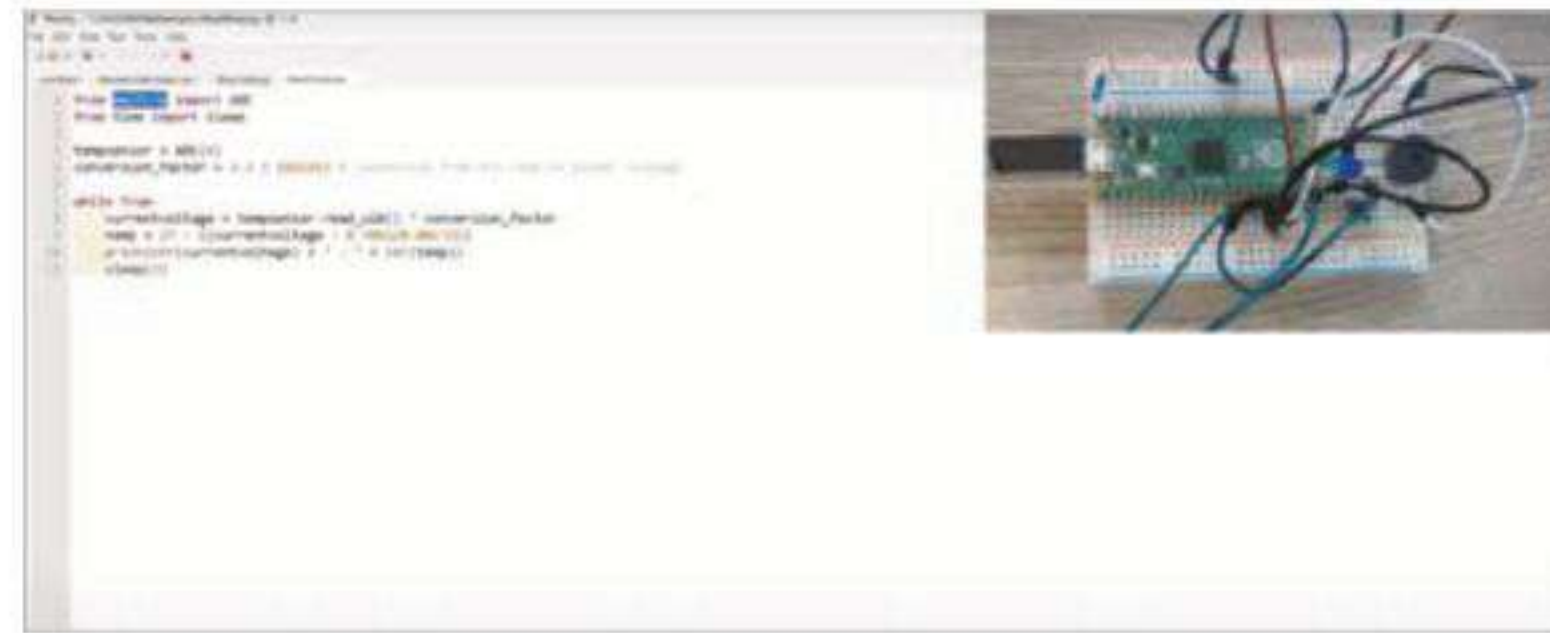
▲ Reading a temperature from Pico
magpi.cc/readtempgit

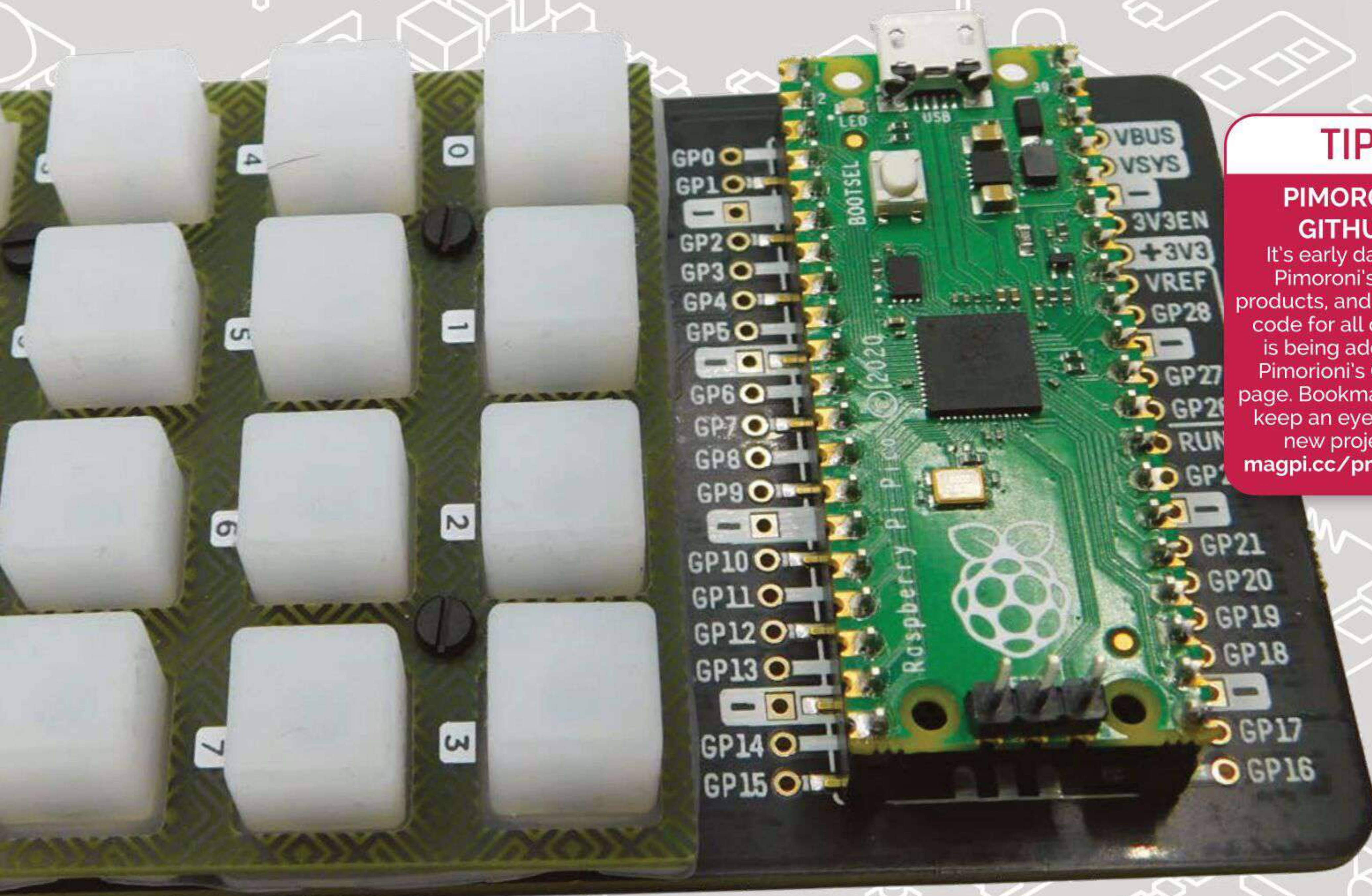▲ Playing music on Pico
magpi.cc/playingmusicgit
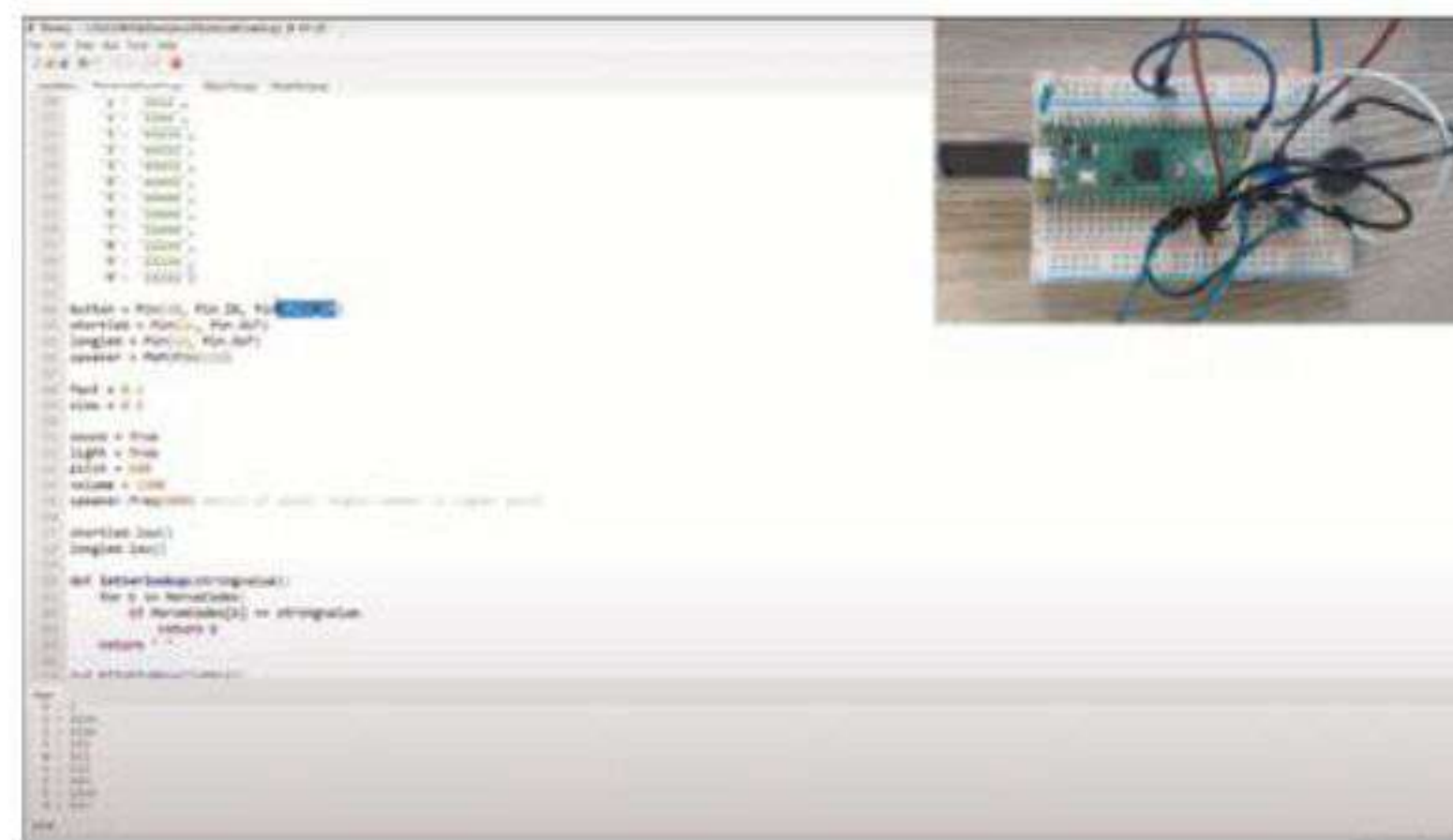
▲ Playing and recording Morse code
magpi.cc/morsecodegit

TIP

PIMORONI GITHUB

It's early days for Pimoroni's Pico products, and example code for all of them is being added to Pimoroni's GitHub page. Bookmark it and keep an eye out for new projects.
magpi.cc/pmpicogit

PICO GAMES
PROJECTS

Get creative with Raspberry Pi Pico gaming

Video games are a staple area of interest for any new computer. At first glance, you might think that Pico doesn't lend itself to games, but you'd be mistaken.

PIO enables a lot of the heavy lifting for the display and input to be taken away from the main controller, freeing the dual-core Arm Cortex M0+ processor to handle the game. The result is a surprisingly versatile system for gaming.

Graham Sanderson has shown that it's possible to emulate a BBC Micro and ZX Spectrum using PIO to output the audio and VGA graphics. There are no instructions just yet, but you can check out the build on YouTube (**magpi.cc/picobbcmicro**).

If you're looking for a build with ready-to-roll code, then take a look at Tetris for Pico, built using the Pico Display Pack (**magpi.cc/picodisplaypack**). It uses the buttons on the latter to rotate and control the pieces, and the small screen displays the game. The code for Pico Tetris can be found on PasteBin (**magpi.cc/picotetris**).

The small screen of the Pico Display Pack or LEDs of the Pico Unicorn Pack lend themselves to Tamagotchi-style games, and you can easily create one for Raspberry Pi Pico using animations. There is the Pixel Pet code for Raspberry Pi Sense HAT on the projects page (**magpi.cc/pixelpet**) that could be easily converted to Raspberry Pi Pico and a Pico Display Pack or Unicorn Pack.

If you are looking for a more complete games system, then Pimoroni's PicoSystem (**magpi.cc/picosystem**) is under development. PicoSystem promises to be a tiny all-in-one portable games console being built around the RP2040 microcontroller at the heart of Pico. It features a small IPS LCD screen, joypad, buttons, and LiPo battery.

Raspberry Pi Pico is proving to be quite the powerhouse for retro gaming, much more so than you'd first imagine. **M**

▲ The BBC Micro computer emulated on Raspberry Pi Pico, with PIO handling the display output

▼ PicoSystem is an all-in-one games console based around the powerful RP2040 microcontroller at the heart of Raspberry Pi Pico

> ❝ If you're looking for a build with ready-to-roll code, then take a look at Tetris for Pico ❞

EASY ELECTRONICS
WITH PICO

Roll your own projects with components and code

Like all Raspberry Pi boards, Pico is excellent for learning electronics and building small circuits that do all manner of things. It's possible to solder wires and components directly to GPIO sockets, but adding a pin header enables you to slot Pico into the sockets of a breadboard (**magpi.cc/breadboard**) and connect and disconnect components without having to solder them.

> ❝ You'll never be short of projects to try out ❞

With pins on Pico, you can follow the 'Getting started with Raspberry Pi Pico' guide (**magpi.cc/gettingstartedpico**). And make sure you pick up a copy of *Get Started with MicroPython on Raspberry Pi Pico* (**magpi.cc/picobook**). You'll also find a fantastic tutorial on using Pico with LEDs and buttons on the following pages.

Easy electronics
Numerous devices have been developed to aid electronic testing and development with Raspberry Pi Pico. SB Components has produced a Pico Breadboard Kit (**magpi.cc/picobread**) that should make prototyping easier. As well as a half-size breadboard, it packs a buzzer, four LEDs, and four push-buttons. It also has dedicated 5V, 3V3, and GND pins.

Another device aiming to make electronics more accessible is the Pico Explorer Base (**magpi.cc/picoexplorer**). This also features a mini breadboard, but one-ups other boards with a 240×240 IPS LCD screen and four buttons (to create interactive menus for your projects). It also has two Breakout Garden slots that you can use to quickly slot in devices from Pimoroni's breakout range (**magpi.cc/pimoronibreakouts**). These include everything from air quality sensors to haptic feedback buttons.

If you want to avoid a nest of wires and experiment with a range of components, check out Seeed's Grove Shield for Raspberry Pi Pico (**magpi.cc/groveshield**). This enables you to plug and play with over 300 Grove modules.

Seeed has provided a walkthrough to using the Grove Shield with Raspberry Pi Pico (**magpi.cc/groveshieldtutorial**) that demonstrates how to attach a buzzer, rotary angle sensor, OLED display, and temperature sensor with Raspberry Pi Pico.

HATs everywhere
Another device is the Raspberry Pi Pico HAT Expansion (**magpi.cc/picohatexpansion**). This is an input/output board that translates the Raspberry Pi Pico pins into the 40-pin header found on Raspberry Pi computers. This enables you to plug any of the myriad of HATs designed for Raspberry Pi into Pico (although it's worth noting that software and APIs designed for Raspberry Pi may need translation work to function in MicroPython or C/C++). If you have a project in Raspberry Pi that's ripe for conversion to Pico, this hardware could be just the thing.

Pimoroni's Pico Omnibus Dual Expander (**magpi.cc/picoomnibus**) enables you to double up on expansion packs, adding two at once. Or, you can use the extra GPIO pins to attach jumper wires or circuitry alongside a Pico Pack. If that's not enough, the Pico Decker Quad Expander (**magpi.cc/picodecker**) goes to town with support for up to four Pico Pack devices. On the board is a 'landing area' with labelled female headers for attaching Pico, and four further landing areas with mirrored headers for attaching add-ons.

One thing's for sure: with Pico you'll never be short of projects to try out. **M**
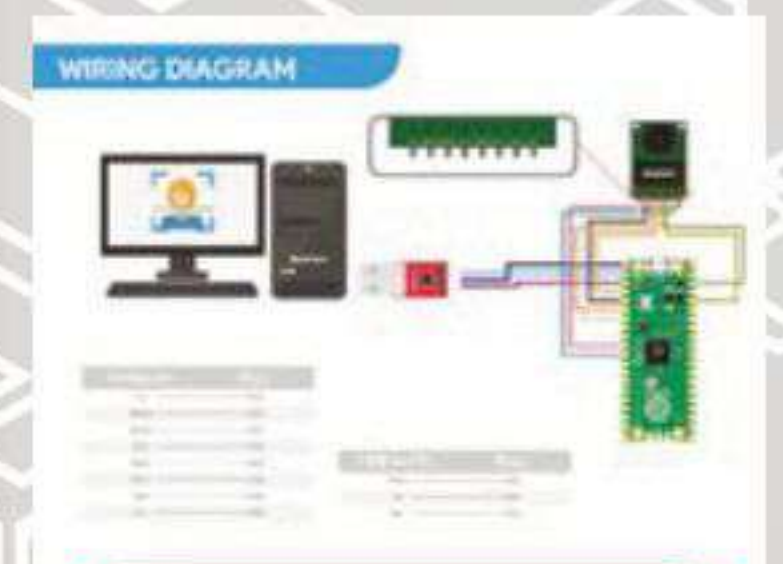
DETECTING PEOPLE WITH AI

Here is one more project that might be suitable for the more advanced makers out there. Kyle from Arducam has got TensorFlow Lite Micro up and running on Raspberry Pi Pico and has created a tutorial on using the machine learning framework for person detection. Connect a camera to your Raspberry Pi Pico and you will be able to use image recognition to detect people. **magpi.cc/tfpico**

▼ The Grove Shield for Raspberry Pi Pico enables you to plug in over 300 modules from the Seeed range
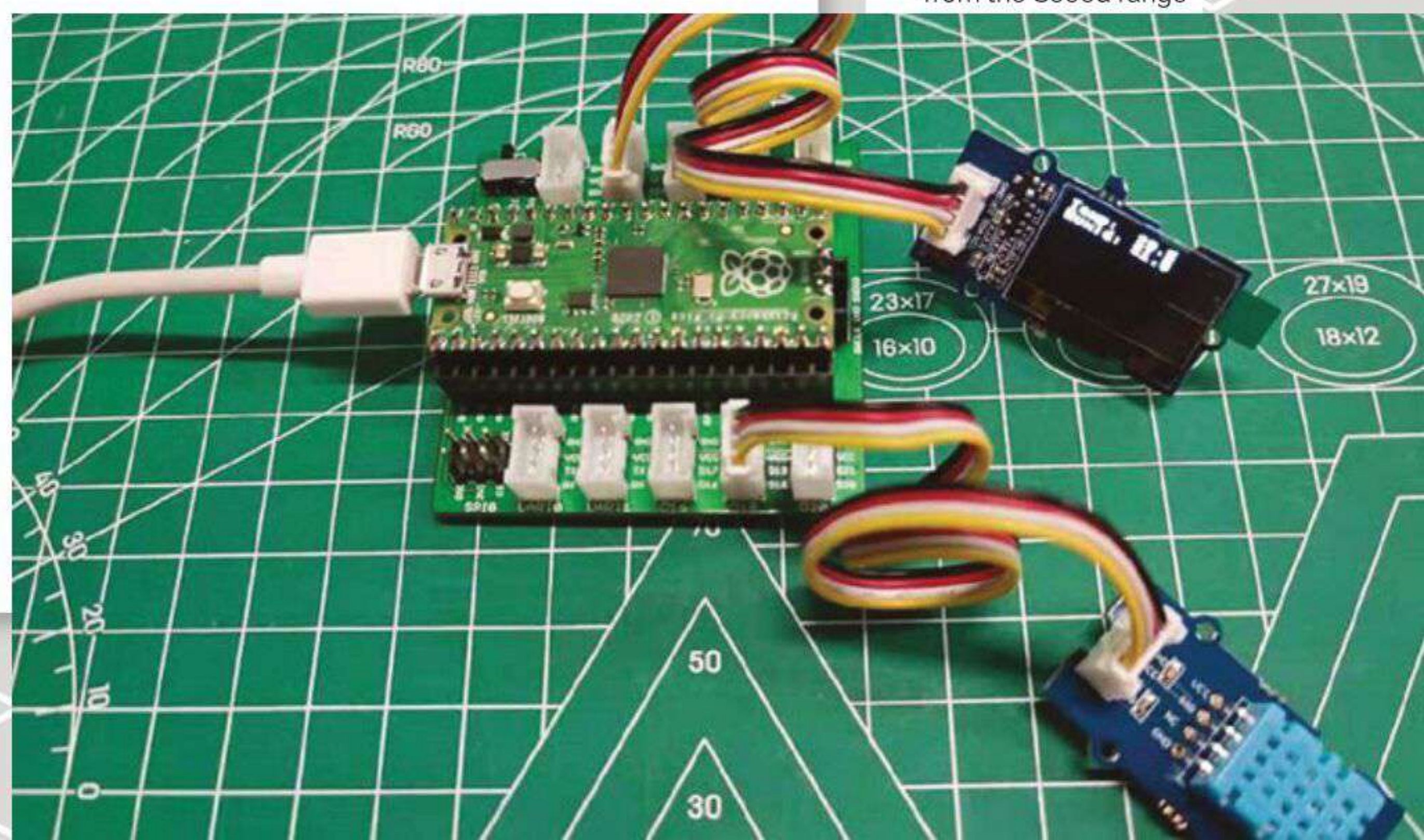
Physical computing
with Raspberry Pi Pico

Start connecting basic electronic components to your Raspberry Pi Pico and writing programs to control and sense them

Raspberry Pi Pico, with its RP2040 microcontroller, is designed with physical computing in mind. Its numerous general-purpose input/output (GPIO) pins let it talk to a range of components, allowing you to build up projects, from lighting LEDs to recording data about the world around you.

Physical computing is no more difficult to learn than traditional computing: if you can do a little Python coding, you'll be able to build your own circuits and program them to do your bidding.

Your first physical computing program: Hello, LED!

Just as printing 'Hello, World' to the screen is a fantastic first step in learning a programming language, making an LED light up is the traditional introduction to learning physical computing. You can get started without any additional components, too: your Raspberry Pi Pico has a small LED, known as a surface-mount device (SMD) LED, on top.

Start by finding the LED: it's the small rectangular component to the left of the micro-USB port at the top of the board (**Figure 1**), marked with a label reading 'LED'. This small LED works just like any other: when it's powered on, it will glow; when it's powered off, it remains dark.

Figure 1

> **Figure 1** The on-board LED is found to the left of the micro-USB connector

The on-board LED is connected to one of RP2040's general-purpose input/output pins, GP25. This is one of the 'missing' GPIO pins present on RP2040 but not broken out to a physical pin on the edge of your Pico. While you can't connect any external hardware to the pin, other than the on-board LED, it can be treated just the same as any other GPIO pin within your programs – and it makes a great way to add an output to your programs without needing any extra components.

Load Thonny and, with Pico connected to your Raspberry Pi, click 'Python' at the bottom-right, then select 'MicroPython (Raspberry Pi Pico)' as the interpreter. Click into the script area, and start your program with the following line:

```
import machine
```

This short line of code is key to working with MicroPython on your Pico: it loads, or *imports*, a collection of MicroPython code known as a *library* – in this case, the 'machine' library. The machine library contains all the instructions MicroPython needs to communicate with the Pico and other MicroPython-compatible devices, extending the language for physical computing. Without this line, you won't be able to control any of your Pico's GPIO pins – and you won't be able to make the on-board LED light up.

The machine library exposes what is known as an *application programming interface* (API). The name sounds complicated, but describes exactly what it does: it provides a way for your program, or the *application*, to communicate with Pico via an interface. The next line of your program provides an example of the machine library's API:

```
led_onboard = machine.Pin(25, machine.Pin.OUT)
```

This line defines an object called `led_onboard`, which offers a friendly name you can use to refer to the on-board LED later in your program. It's technically possible to use any name here – like susan, gupta, or fish_sandwich – but it's best to stick with names which describe the variable's purpose, to make the program easier to read and understand.

The second part of the line calls the Pin function in the machine library. This function, as its name suggests, is designed for handling your Pico's GPIO pins. At the moment, none of the GPIO pins – including GP25, the pin connected to the on-board LED – knows what they're supposed to be doing. The first argument, `25`, is the number of the pin you're setting up; the second, `machine.Pin.OUT`, tells Pico the pin should be used as an output rather than an input.

That line alone is enough to set the pin up, but it won't light the LED. To do that, you need to tell Pico to actually turn the pin on. Type the following code on the next line:

```
led_onboard.value(1)
```

" The on-board LED is connected to one of RP2040's GPIO pins "

It might not look it, but this line is also using the machine library's API. Your earlier line created the object `led_onboard` as an output on pin GP25; this line takes the object and sets its value to 1 for 'on' – it could also set the value to 0, for 'off'.

Click the Run button and save the program on your Pico as **Blink.py**. You'll see the LED light up. Congratulations – you've written your first physical computing program!

You'll notice, however, the LED stays lit: that's because your program tells Pico to turn it on, but never tells it to turn it off. You can add another line at the bottom of your program:

```
led_onboard.value(0)
```

Run the program this time, though, and the LED never seems to light up. That's because Pico works very, very quickly – far more quickly than you can see with the naked eye. The LED is lighting up, but

Figure 2

All the holes in each rail are connected

The five holes in each row are connected

▲ **Figure 2** The internal connections on a breadboard
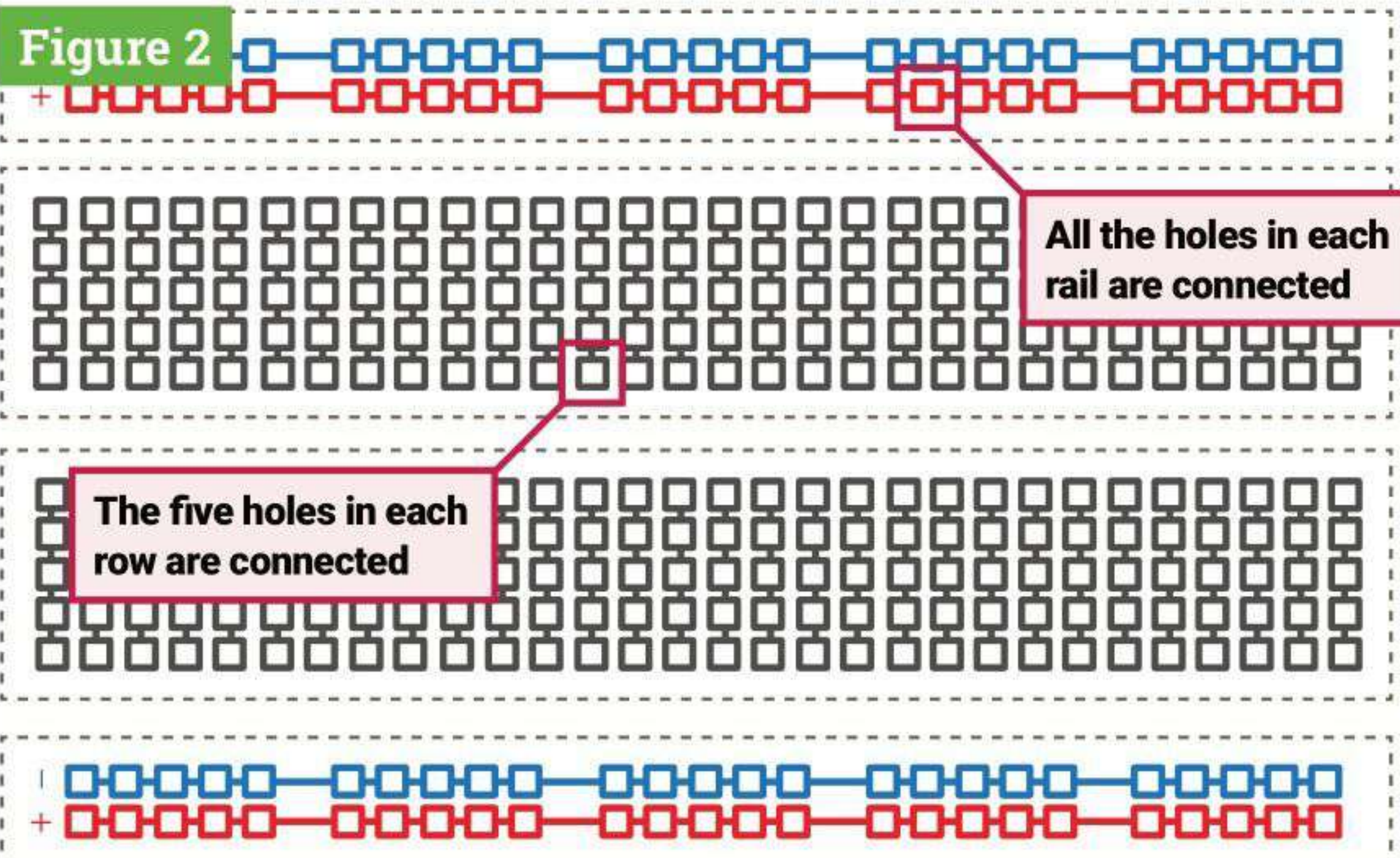
for such a short time it appears to remain dark. To fix that, you need to slow your program down by introducing a *delay*.

Go back to the top of your program: click at the end of the first line and press **ENTER** to insert a new second line. On this line, type:

```
import utime
```

Like `import machine`, this line imports a new library into MicroPython: the 'utime' library. This library handles everything to do with time, from measuring it to inserting delays into your programs.

Go to the bottom of your program, and click on the end of the line `led_onboard.value(1)`, then press **ENTER** to insert a new line. Type:

```
utime.sleep(5)
```

This calls the `sleep` function from the utime library, which makes your program pause for whatever number of seconds you typed – in this case, five seconds.

Click the Run button again. This time you'll see the on-board LED on your Pico light up, stay lit for five seconds – try counting along – and go out again.

Finally, it's time to make the LED blink. To do that, you'll need to create a loop. Rewrite your program so it matches the one below:

```
import machine
import utime

led_onboard = machine.Pin(25, machine.Pin.OUT)
```
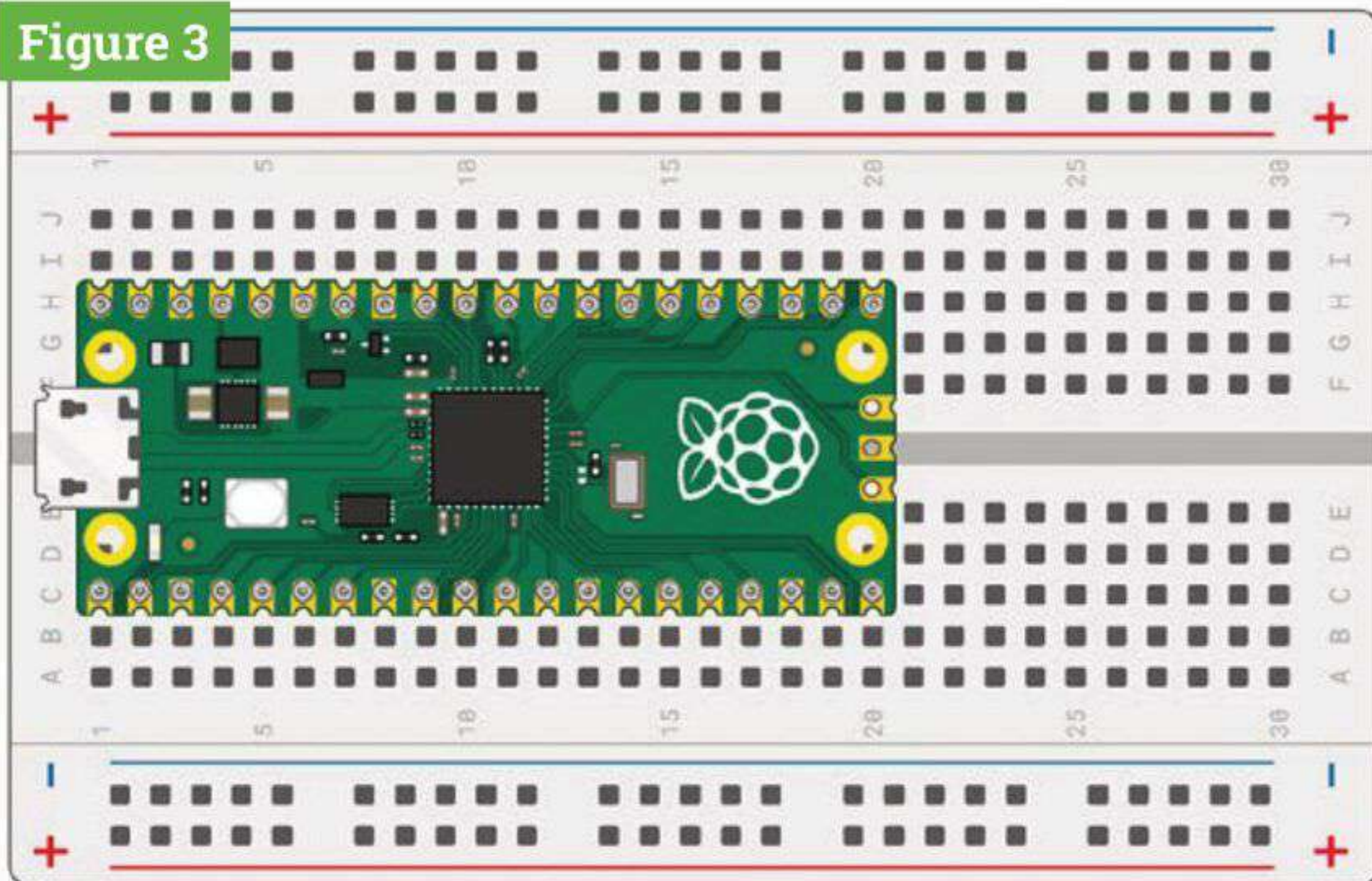
TIP

PIN NUMBERS
When talking about the GPIO pins on your Pico they're usually referred to using their full names: GP25 for the pin connected to the on-board LED, for example. In MicroPython, though, the letters G and P are dropped – so make sure you write '25' rather than 'GP25' in your program, or it won't work!

Figure 3

▲ **Figure 3** Your Pico is designed to sit securely in a solderless breadboard

```
while True:
    led_onboard.value(1)
    utime.sleep(5)
    led_onboard.value(0)
    utime.sleep(5)
```

Remember that the lines inside the loop need to be indented by four spaces, so MicroPython knows they form the loop. Click the Run icon again and you'll see the LED switch on for five seconds, switch off for five seconds, and switch on again – constantly repeating in an infinite loop. The LED will continue to flash until you click the Stop icon to cancel your program and reset your Pico.

There's another way to handle the same job, too: using a *toggle*, rather than setting the LED's output to 0 or 1 explicitly. Delete the last four lines of your program and replace them so it looks like this:

```
import machine
import utime

led_onboard = machine.Pin(25, machine.Pin.OUT)

while True:
    led_onboard.toggle()
    utime.sleep(5)
```

Run your program again. You'll see the same activity as before: the on-board LED will light up for five seconds, then go out for five seconds, then light up again in an infinite loop. This time, though, your program is two lines shorter: you've *optimised* it. Available on all digital output pins, `toggle()` simply switches between on and off: if the pin is currently on, `toggle()` switches it off; if it's off, `toggle()` switches it on.

TIP

UTIME VS TIME
If you've programmed in Python before, you'll be used to using the 'time' library. The utime library is a version designed for microcontrollers like the Pico – the 'u' standing for 'μ', the Greek letter 'mu', which is used as a shorthand for 'micro'. If you forget and use `import time`, don't worry: MicroPython will automatically use the utime library instead.

Using a breadboard

The following projects in this chapter will be much easier to complete if you're using a breadboard to hold the components and make the electrical connections.

A breadboard (**Figure 2**) is covered with holes – spaced, to match components, 2.54 mm apart. Under these holes are metal strips which act like the jumper wires you've been using until now. These run in rows across the board, with most boards having a gap down the middle to split them in two halves.

Many breadboards also have letters across the top and numbers down the sides. These allow you to find a particular hole: A1 is the top-left corner, B1 is the hole to the immediate right, while B2 is one hole down from there. A1 is connected to B1 by the hidden metal strips, but no hole marked with a 1 is ever connected to any hole marked with a 2 unless you add a jumper wire yourself.

> ❝ Adding electronic components to a breadboard is simple ❞

Larger breadboards also have strips of holes down the sides, typically marked with red and black or red and blue stripes. These are the power *rails*, and are designed to make wiring easier: you can connect a single wire from one of your Pico's ground pins to one of the power rails – typically marked with a blue or black stripe and a minus symbol – to provide a common ground for lots of components on the breadboard, and you can do the same if your circuit needs 3.3 V or 5 V power. Note: All holes joined by a stripe are connected; a gap indicates a break.

Adding electronic components to a breadboard is simple: just line their leads (the sticky-out metal parts) up with the holes and gently push until the component is in place. For connections you need to make beyond those the breadboard makes for you, you can use male-to-male (M2M) jumper wires; for connections from the breadboard to external devices, like your Raspberry Pi Pico, use male-to-female (M2F) jumper wires.

Never try to cram more than one component lead or jumper wire into any single hole on the breadboard. Remember: holes are connected in rows, aside from the split in the middle, so a component lead in A1 is electrically connected to anything you add to B1, C1, D1, and E1.

Push your Pico into the breadboard so it straddles the middle gap and the micro USB port is at the very top of the board (see **Figure 3**). The top-left pin, Pin 0, should be in the breadboard row marked with a 1, if your breadboard is numbered. Before pushing your Pico down, make sure the header pins are all properly positioned – if you bend a pin, it can be difficult to straighten it again without it breaking.

Gently push the Pico down until the plastic parts of the header pins are touching the breadboard. This means the metal parts of the header pins are fully inserted and making good electrical contact with the breadboard.

Next steps: an external LED

So far, you've been working with your Pico on its own – running MicroPython programs on its RP2040 microcontroller and toggling the on-board LED on and off. Microcontrollers are usually used with *external* components, though – and your Pico is no exception.

For this project, you'll need a breadboard, male-to-male (M2M) jumper wires, an LED, and a 330 Ω resistor – or as close to 330 Ω as you have available. If you don't have a breadboard, you can use female-to-female (F2F) jumper wires, but the circuit will be fragile and easy to break.

Hold the LED in your fingers: you'll see one of its leads is longer than the other. The longer lead is known as the *anode*, and represents the positive side of the circuit; the shorter lead is the *cathode*, and represents the negative side. The anode needs to be connected to one of your Pico's GPIO pins via the resistor; the cathode needs to be connected to a ground pin.

Start by connecting the resistor: take one end (it doesn't matter which) and insert it into the breadboard in the same row as your Pico's GP15 pin at the bottom-left – if you're using a numbered breadboard with your Pico inserted at the very top, this should be row 20. Push the other end into a free row further down the breadboard – we're using row 24.

Take the LED, and push the longer leg – the anode – into the same row as the end of the resistor. Push the shorter leg – the cathode – into the same row but across the centre gap in the breadboard, so it's lined up but not electrically connected to the longer leg except through the LED itself. Finally, insert a male-to-male (M2M) jumper wire into the same row as the shorter leg of the LED, then either connect it directly to one of your Pico's ground pins (via another hole in its row) or to the negative side

of your breadboard's power rail. If you connect it to the power rail, finish the circuit by connecting the rail to one of your Pico's ground pins. Your finished circuit should look like **Figure 4**.

Controlling an external LED in MicroPython is no different to controlling your Pico's internal LED: only the pin number changes. If you closed Thonny, reopen it and load your **Blink.py** program from earlier in the chapter. Find the line:

```
led_onboard = machine.Pin(25, machine.Pin.OUT)
```
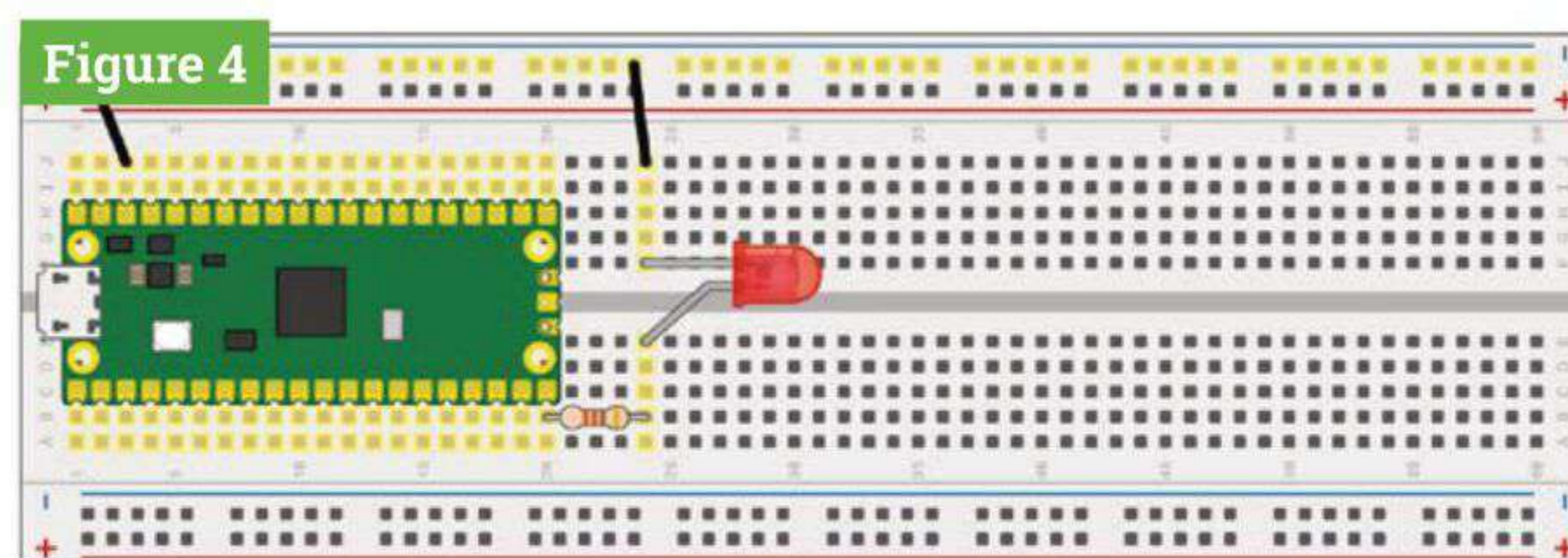
Edit the pin number, changing it from 25 – the pin connected to your Pico's internal LED – to 15, the pin to which you connected the external LED. Also edit the name you created: you're not using the on-board LED any more, so have it say `led_external` instead. You'll also have to change the name elsewhere in the program, until it looks like this:

```
import machine
import utime

led_external = machine.Pin(15, machine.Pin.OUT)

while True:
    led_external.toggle()
    utime.sleep(5)
```

▼ **Figure 4** The finished circuit, with an LED and a resistor

Figure 4

TIP

RESISTANCE IS VITAL

The resistor is a vital component in this circuit: it protects both your Pico and the LED by limiting the amount of electrical current the LED can draw. Without it, the LED can pull too much current and burn itself – or your Pico – out. When used like this, the resistor is known as a current-limiting resistor. The exact value of resistor you need depends on the LED you're using, but 330 Ω works for most common LEDs. The higher the value, the dimmer the LED; the lower the value, the brighter the LED.

Never connect an LED to your Pico without a current-limiting resistor, unless you know the LED has a built-in resistor of appropriate value.

Inputs: reading a button

Outputs like LEDs are one thing, but the 'input/output' part of 'GPIO' means you can use pins as inputs too. For this project, you'll need a breadboard, male-to-male jumper wires, and a push-button switch. If you don't have a breadboard, you can use female-to-female (F2F) jumper wires, but the button will be much harder to press without accidentally breaking the circuit.

Remove any other components from your breadboard except your Pico, and begin by adding the push-button switch. If your push-button has only two legs, make sure they're in different-numbered rows on the breadboard somewhere below your Pico. If it has four legs, turn it so the sides the legs come from are along the breadboard's rows and the flat leg-free sides are at the top and bottom before pushing it home so it straddles the centre divide of the breadboard.

Connect the positive power rail of your breadboard to your Pico's 3V3 pin, and from there to one of the legs of the switch; then connect the other leg to pin GP14 on your Pico – it's the one just above the pin you used for the LED project, and should be in row 19 of your breadboard.

" Pushing the button has completed the circuit and changed the value read "

If you're using push-button with four legs, your circuit will only work if you use the correct pair of legs: the legs are connected in pairs, so you need to either use the two legs on the same side or (as seen in **Figure 5**) diagonally opposite legs.

Load Thonny, if you haven't already, and start a new program with the usual line:

```
import machine
```

▼ Figure 5 Wiring a four-leg push-button switch to GP14

Next, you need to use the machine API to set up a pin as an input, rather than an output:
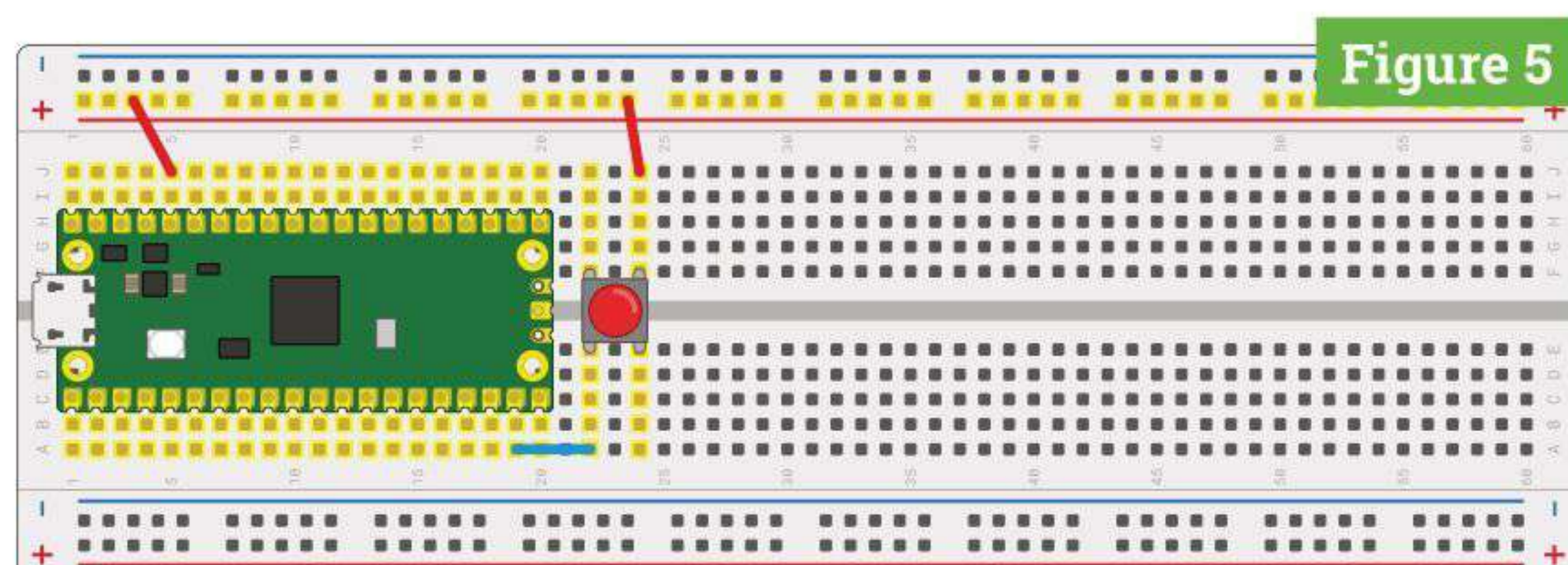
Figure 5

```
button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)
```

This works in the same way as your LED projects: an object called 'button' is created, which includes the pin number – GP14, in this case – and configures it as an input with the resistor set to pull-down. Creating the object, though, doesn't mean it will do anything by itself – just as creating the LED objects earlier didn't make the LEDs light up.

To actually read the button, you need to use the machine API again – this time using the `value` function to read, rather than set, the value of the pin. Type the following line:

```
print(button.value())
```

Click on Run and save your program as **Button.py** – remembering to make sure it saves on Raspberry Pi Pico. Your program will print out a single number: the value of the input on GP14. Because the input is using a pull-down resistor, this value will be 0 – letting you know the button isn't pushed.

Hold down the button with your finger, and press the Run icon again. This time, you'll see the value 1 printed to the Shell: pushing the button has completed the circuit and changed the value read from the pin. To read the button continuously, you'll need to add a loop to your program. Edit the program so it reads as below:

```
import machine
import utime

button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)

while True:
    if button.value() == 1:
        print("You pressed the button!")
        utime.sleep(2)
```

Click the Run button again. This time, nothing will happen until you press the button; when you do, you'll see a message printed to the Shell area. The delay, meanwhile, is important: remember, your Pico runs a lot faster than you can read, and without the delay even a brief press of the button can print hundreds of messages to the Shell!

You'll see the message print every time you press the button. If you hold the button down for longer than the two-second delay, it will print the message every two seconds until you let go of the button.

Inputs and outputs: putting it all together

Most circuits have more than one component, which is why your Pico has so many GPIO pins. It's time to put everything you've learned together to build a more complex circuit: a device which switches an LED on and off with a button.

In effect, this circuit combines both of the previous circuits into one. You may remember you used pin GP15 to drive the external LED, and pin GP14 to read the button; now rebuild your circuit so both the LED and the button are on the breadboard at the same time, still connected to GP15 and GP14 (see **Figure 6**). Don't forget the current-limiting resistor for the LED!

Start a new program in Thonny, and begin importing the two libraries your program will need:

```
import machine
import utime
```

Next, set up both the input and output pins:

```
led_external = machine.Pin(15, machine.Pin.OUT)
button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)
```

Then create a loop which reads the button:

```
while True:
    if button.value() == 1:
```

Rather than printing a message to the Shell, though, this time you're going to toggle the output pin and the LED connected to it based on the value of the input pin. Type the following, remembering it will need to be indented by eight spaces – which Thonny should have automatically handled when you pressed **ENTER** at the end of the line above:

```
        led_external.value(1)
        utime.sleep(2)
```

That's enough to turn the LED on, but you'll also need to turn it off again when the button isn't being pressed. Add the following new line, using the **BACKSPACE** key to delete four of the eight spaces – meaning the line will not be part of the if statement, but will form part of the infinite loop:

```
    led_external.value(0)
```

Your finished program should look like this:

Figure 6

```
import machine
import utime


led_external = machine.Pin(15, machine.Pin.OUT)
button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)

while True:
    if button.value() == 1:
        led_external.value(1)
        utime.sleep(2)
    led_external.value(0)
```

▲ **Figure 6** The finished circuit, with both a button and an LED

Click on Run and save the program as **Switch.py** on your Pico. At first, nothing will happen; push the button, though, and you'll see the LED light up. Let go of the button; after two seconds, the LED will go out again until you press the button again.

Congratulations: you've built your first circuit which controls one pin based on the input from another – a building block for bigger things! M

SmartiPi Touch Pro

▶ SmartiCase ▶ **magpi.cc/smartipi** ▶ £30 / $35

The latest addition to the SmartiPi Touch family brings improved design, cooling, and a wealth of options. **PJ Evans** looks into the case

We're big fans of the SmartiCase range here at *The MagPi* magazine. The SmartiPi Touch 2 was a big hit in 2020, with a solid 8/10 awarded. Now it's back with a third iteration, the SmartiPi Touch Pro.

All models mount the official 7-inch Raspberry Pi touchscreen and a Raspberry Pi of your choice into a single case running from one power supply. The result is a small, freestanding unit, perfect for kiosk-style applications. Add a keyboard and monitor for a small, but perfectly formed, workstation.

The Touch Pro is a solid refinement of its predecessor. Although similar in appearance, the optional camera mount has been moved to the base of the screen, which gives it a slimmer profile. The internals have been redesigned to create more space: in fact, you can fit two HAT form-factor devices side by side. Cooling has been improved with a small optional fan mounted on rubber pillars to reduce vibration.

Construction was simple and completed in about 20 minutes, thanks to a well-written online guide. It's even easier than the previous models. Everything you need is included, along with options for fan covers, a range of port blockers

> ❝ The Touch Pro is a solid refinement of its predecessor ❞

for both Raspberry Pi 3 and 4 configurations, ribbon cables for the display and camera, and – very neatly – a Y-adapter for both USB micro and USB C that now mounts inside the case to give a smart single connector to run both the device and

▲ Multiple mounting options make this case suitable for both home and business applications

screen. We also received the metal base accessory (sold separately) which gives the assembly a solid footing; your cat would struggle to topple this.

Room for everything

Space is a common frustration in Raspberry Pi cases, and it is addressed head-on with the SmartiPi Touch Pro. There is a choice of two rear covers, one with 25 mm clearance above the Raspberry Pi, and a larger version with a whopping 45 mm to play with. Even with the standard header, you can get a low-profile HAT mounted. If you can use jumper cables, you can even mount another HAT alongside. With the larger enclosure, even the larger HATs on the market won't be constricted.

Industrial applications have also been considered. A 'stealth' mode allows Raspberry Pi to be mounted fully inside, giving no easy access to the ports. And if wall or arm mounting would make for a cool touchscreen controller, the rear of the case features VESA mounting and eyelets for hanging from screws. If you want a custom base, additional hinges are provided that can be screwed on to your mount of choice.

We were impressed by the build quality, especially at the very reasonable price point. This SmartiPi Touch Pro has been carefully thought through and

customer feedback considered. It's a solid injection-moulded construction riddled with cut-outs so you can customise to your heart's content. Access to the microSD card slot would be nice, but it's blocked by the display ribbon cable. Nevertheless, if you're looking for a kiosk or control-centre project, or even a highly portable computer (as we featured in *The MagPi* issue #98), the SmartiPi Touch Pro is a great bit of kit. M

▲ This smart case makes a perfect control deck or mini workstation

▼ Smart design means lots of space for HATs and more

Verdict

The sheer range of options, spacious interior, solid design, and a great price make this an essential purchase for any touchscreen project, whether it's a control centre or mini desktop.

9/10

RasPad 3

SPECS

SCREEN:
10.1-inch
1280×800
pixel, ten-point
multitouch

DIMENSIONS / WEIGHT:
26×17×4.8 cm;
2 kg

PORTS:
Ethernet, HDMI,
3.5 mm audio,
3 × USB 3.0,
microSD

▼ RasPad OS makes Raspberry Pi OS touch-friendly, adds support for the rotating screen, and provides an on-screen keyboard

This chunky tablet promises touchscreen interaction with Raspberry Pi. **Lucy Hattersley** gives it a tap

RasPad 3 improves upon previous iterations with a 10.1-inch, ten-point multitouch display. Inside, Raspberry Pi 4 sits between two daughterboards. The 'main board' breaks out the Ethernet, USB C, and audio sockets and provides a full-sized HDMI port for a second external display and an RCA power socket (a 15 V 2 A AC adapter is included as power supply).

The smaller 'microSD card and button board' connects to the microSD port on Raspberry Pi and enables three buttons (power, plus brightness controls).

Inside the large wedge at the bottom sits a three-cell 3 Ah lithium battery (we got two-and-a-half hours of use).

One notable absence from the case is GPIO pins. However, a small gap in the case enables you to feed a ribbon cable to extend the GPIO header.

We found assembly easy. Use four screws to affix Raspberry Pi 4 to the case, then use the USB cables, micro HDMI cables, and Type C to connect Raspberry Pi 4 to the main board. FFC cables are used to connect the smaller daughterboard to Raspberry Pi. These are easy to connect, but the instructions do not mention how to gently pull out the connector and push them back in to lock the cable.

> ❝ RasPad 3 is terrific for demystifying the technology that underpins tablets ❞

Three small heatsinks are attached to the Raspberry Pi and a fan, screwed in place to the bottom half of the case. Finally, a neat touch. A small Accelerometer SHIM Module is placed on top of the GPIO pins on Raspberry Pi. When running the RasPad OS, this enables a rotating display. Four more screws are used to assemble the case. It's important not to leave the microSD card inserted when assembling or disassembling the case, as it will (and indeed did) break.

Custom OS

RasPad OS (**magpi.cc/raspados**) is based on the latest Raspberry Pi OS but with a refreshed interface with larger, touch-friendly buttons, additional software support, and tablet-friendly features: there's an on-screen keyboard and support for the aforementioned accelerometer.

One aspect of RasPad 3 that disappoints straight out of the box is the built-in fan (which you will quickly remove). We've never encountered a

▲ Discover the technology that sits inside a tablet with RasPad

Raspberry Pi product that makes such a persistent noise. It's a constant buzzing and we found no fan throttling controls in software or hardware.

We found the fan intolerable, to the point where we reopened the case and removed it, and dug out our heat testing setup to see what performance was like without it (**magpi.cc/stresstest**). We measured the idle baseline temperature at 65°C and it ran at full stress for several minutes before hitting the 80°C mark (where Raspberry Pi OS starts to throttle back performance). We found little difference to using a Raspberry Pi in the official case. As usual, we see no reason for a fan to be used with Raspberry Pi, unless you plan to overclock. Once the fan was stripped out, we were able to appraise RasPad 3 with kinder eyes.

As a tablet, it functions well. The screen is nice to look at, and touchscreen performance is snappy and responds quickly (if a little haphazardly). While functional, the on-screen keyboard is too small for our fingers and a chore to type on. Still, you can add a Bluetooth or wired keyboard for

more detailed work. It's chunky, but you can hold RasPad in your hands and rotate it around like a commercial tablet. While it's on a surface, the wedge provides two distinct viewing angles. You do lose the ability to use the touchscreen when a second monitor is attached, but it performs admirably as a smaller second display.

RasPad 3 is terrific for demystifying the technology that underpins tablets (key technology in many younger learners' lives). It may be chunky, but you can open it up and see the processor, screen, battery, and accelerometer in action. It may not be as slick as a commercial tablet, but the learning process is more rewarding.

As a daily device, things are less impressive. It's painful to watch RasPad 3 side-by-side against pi-top's FHD Touch Display and Bluetooth keyboard (**magpi.cc/pitopfhd**). The extra money spent on the pi-top is well worth it.

Meanwhile, at the entry end of the scale, devices such as SmartiPi Pro offer a touchscreen display setup at a much more affordable cost. 🅼

Verdict

It's easy and fun to set up RasPad 3, but once the tablet components lesson is over, it's not great fun to use. Jarring elements (in particular the fan) don't help. There are better Raspberry Pi 4 tablet and laptop options on the market.

6/10

10 Amazing:

Raspberry Pi Pico add-ons

Upgrades for the newest
Raspberry Pi family member

Raspberry Pi Pico has been out for only six weeks and it's already got a huge host of add-ons specifically made for it! Some of these have really inspired us to think up new projects. Here are just ten of them. **M**
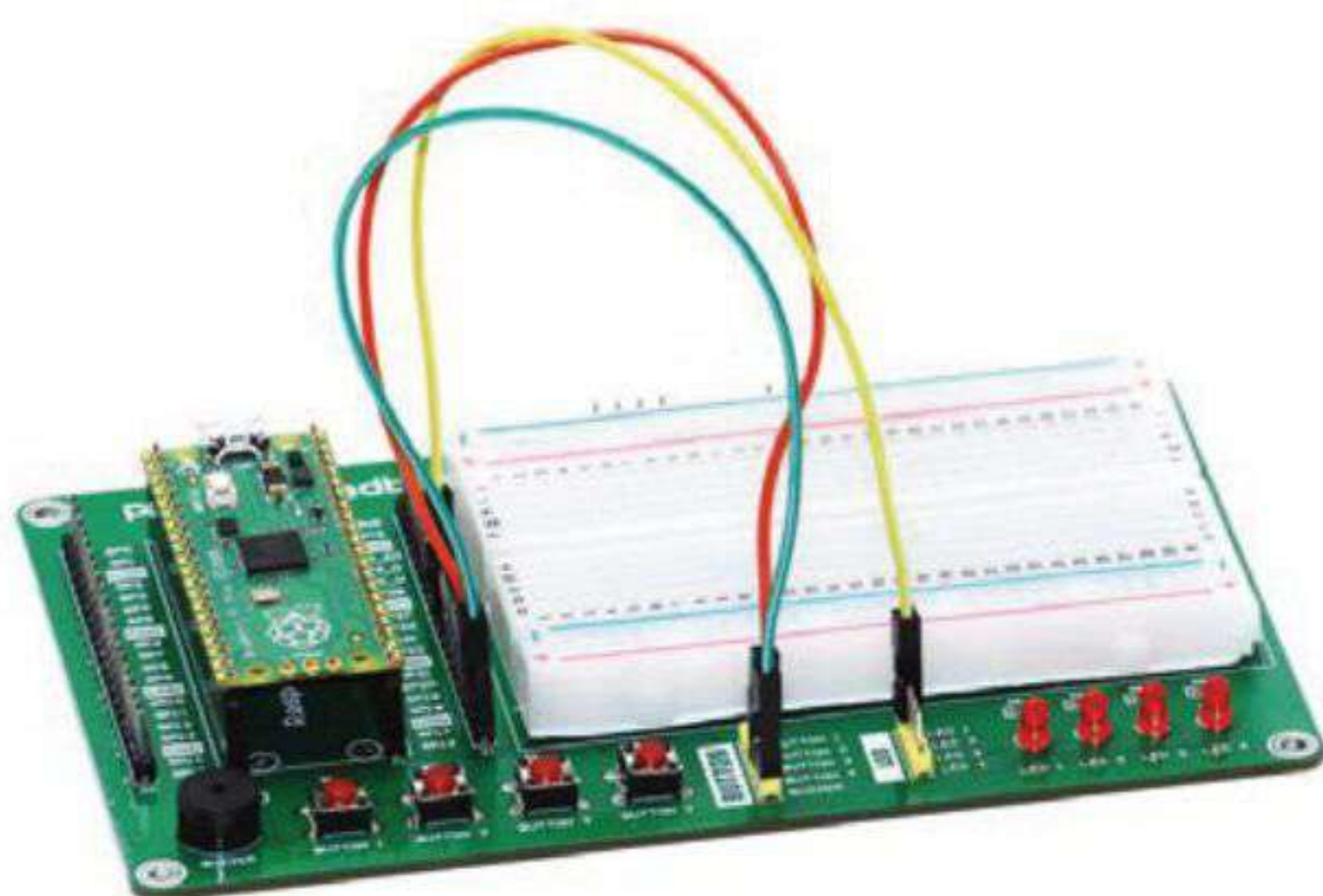
▲ Breadboard Kit

Prototyping board

Another way of learning is by having components connected straight to the same breadboard as Pico itself. The breadboard here is sizeable and the built-in buttons and LEDs are a nice addition.

£14 / $20 | **magpi.cc/picobread**
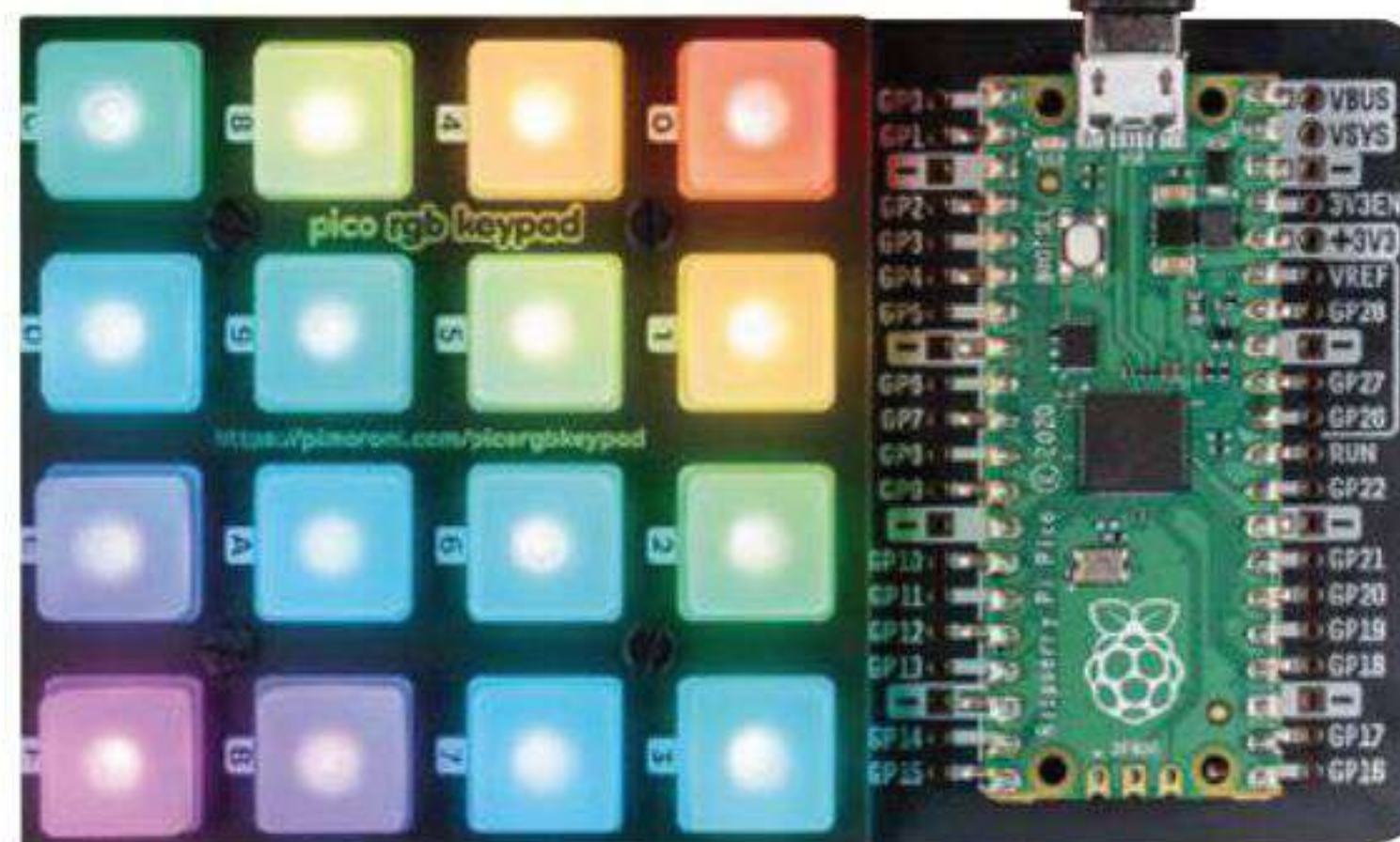
▲ RGB Keypad Base

Customisable keypad

We like the keypad range Pimoroni does, and we're already thinking up ways to create our own macro keypad with one of these and a Pico. Look out for that in a future issue.

£22 / $25 | **magpi.cc/rgbkeypad**

▶ Get Started with MicroPython on Raspberry Pi Pico

Book of knowledge

The ultimate accessory for Pico is the *Get Started with MicroPython* book, which has a breakdown of what Pico can do, and how you can make use of its various abilities.

£10 / $14 (or free PDF) | **magpi.cc/picobook**
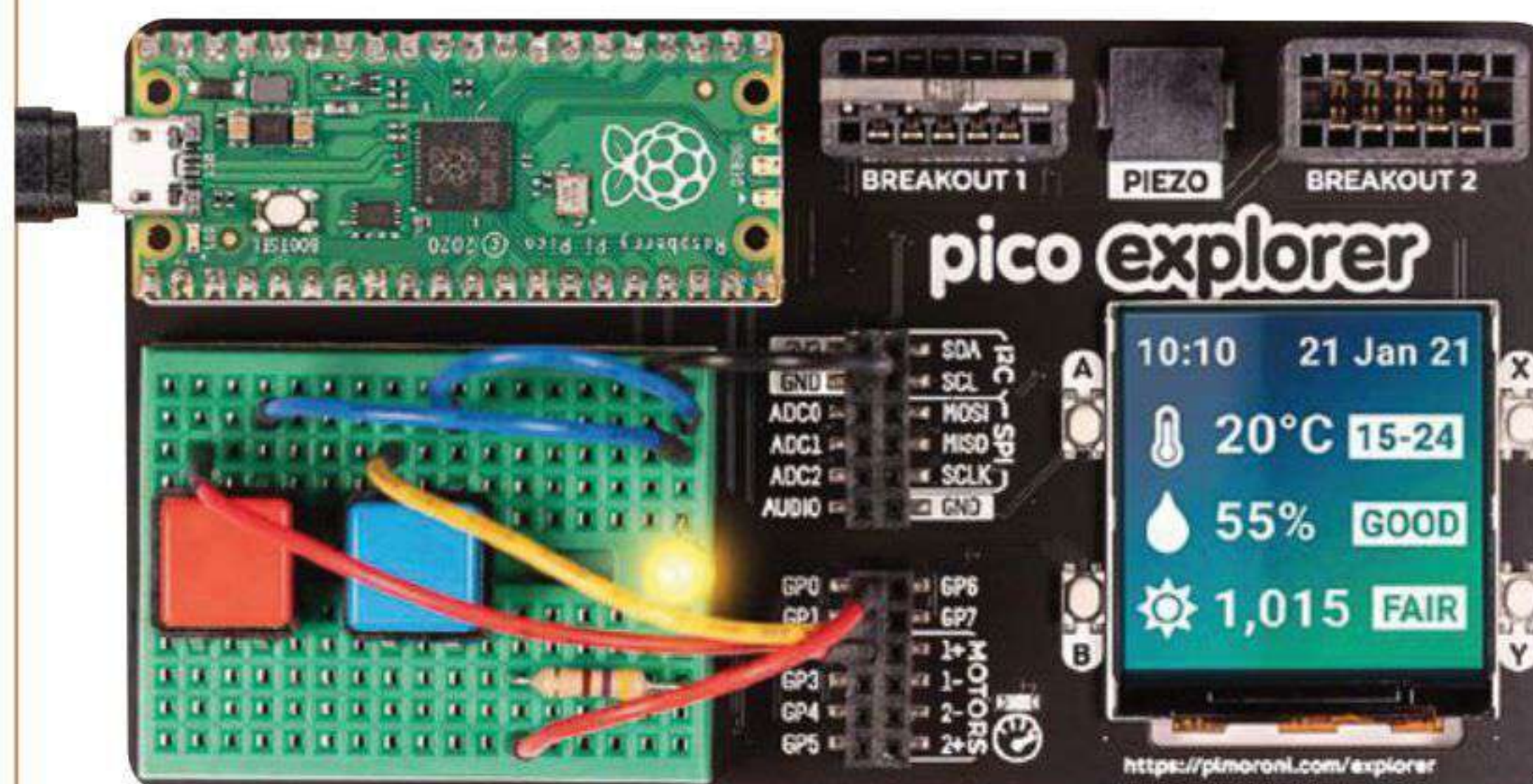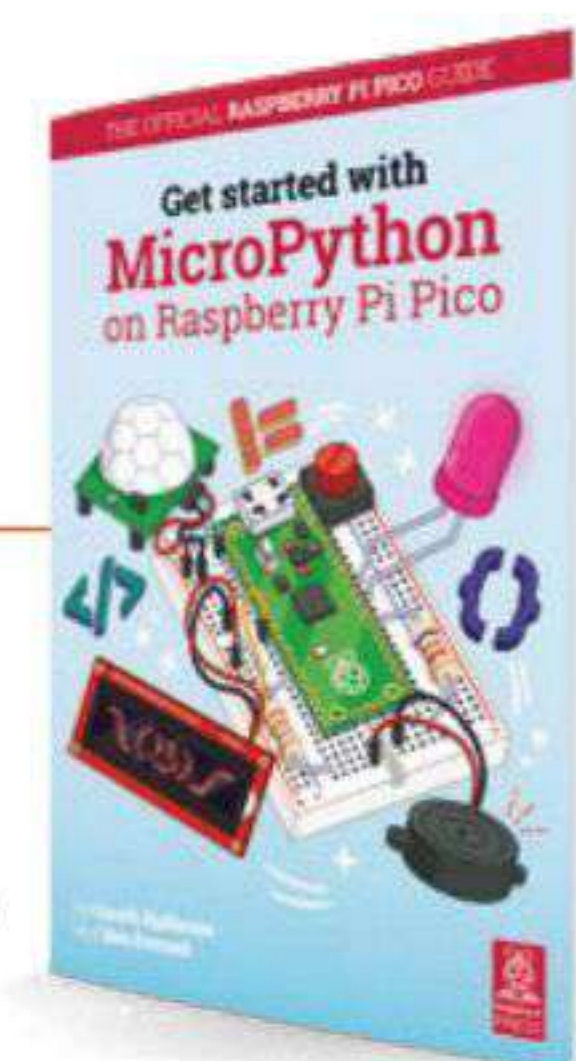
▲ Explorer Base

Electronics with Pico

These types of breadboard breakout boards always rate highly with us as simple and easy ways to get to grips with how tech works. While others work well with a Raspberry Pi, this one is perfect for Pico.

£22 / $26 | **magpi.cc/explorerbase**
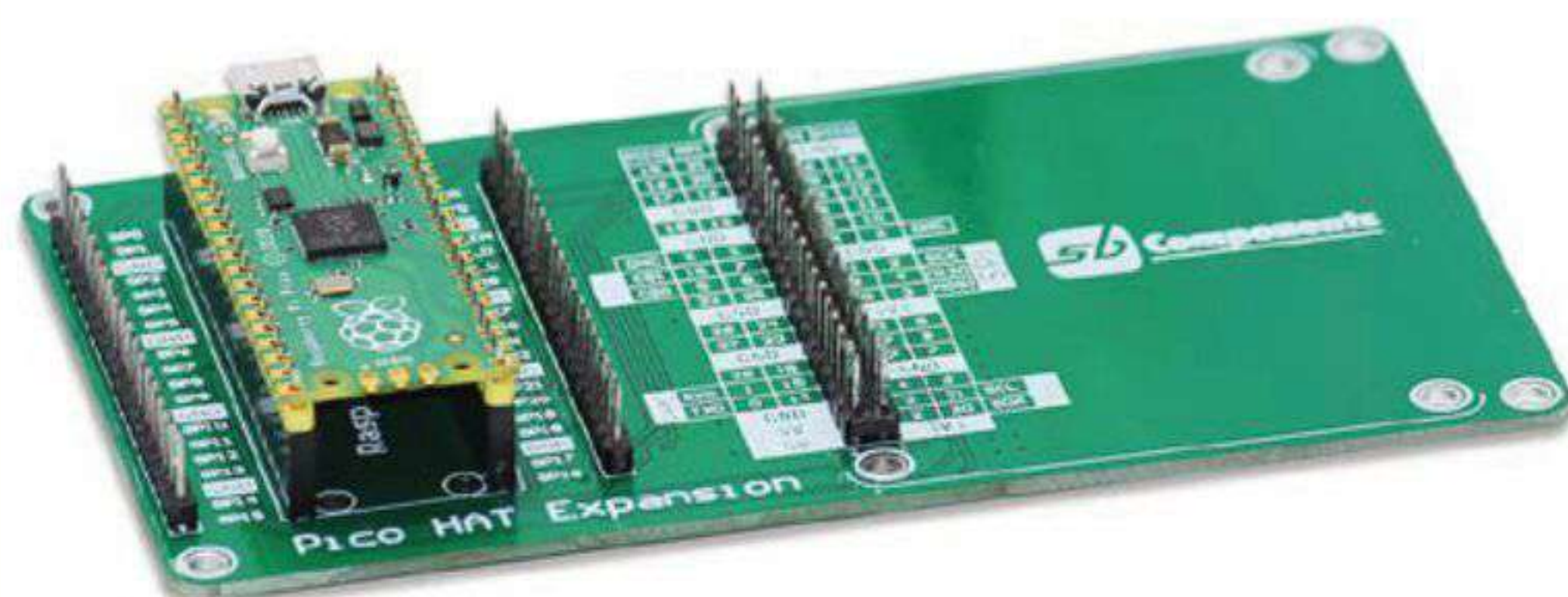
▲ HAT Expansion

Add HATs

Fancy using a Raspberry Pi HAT with a Pico? SB Components has you covered with this special add-on. It also keeps the pins on Pico available to use.

£10 / $15 | **magpi.cc/picohat**

▶ Female headers

Opposite header style

If you want pins that you can plug jumper wires into, or any other kind of wire, these female headers will be right up your alley.

£1 / $2 | **magpi.cc/femheader**
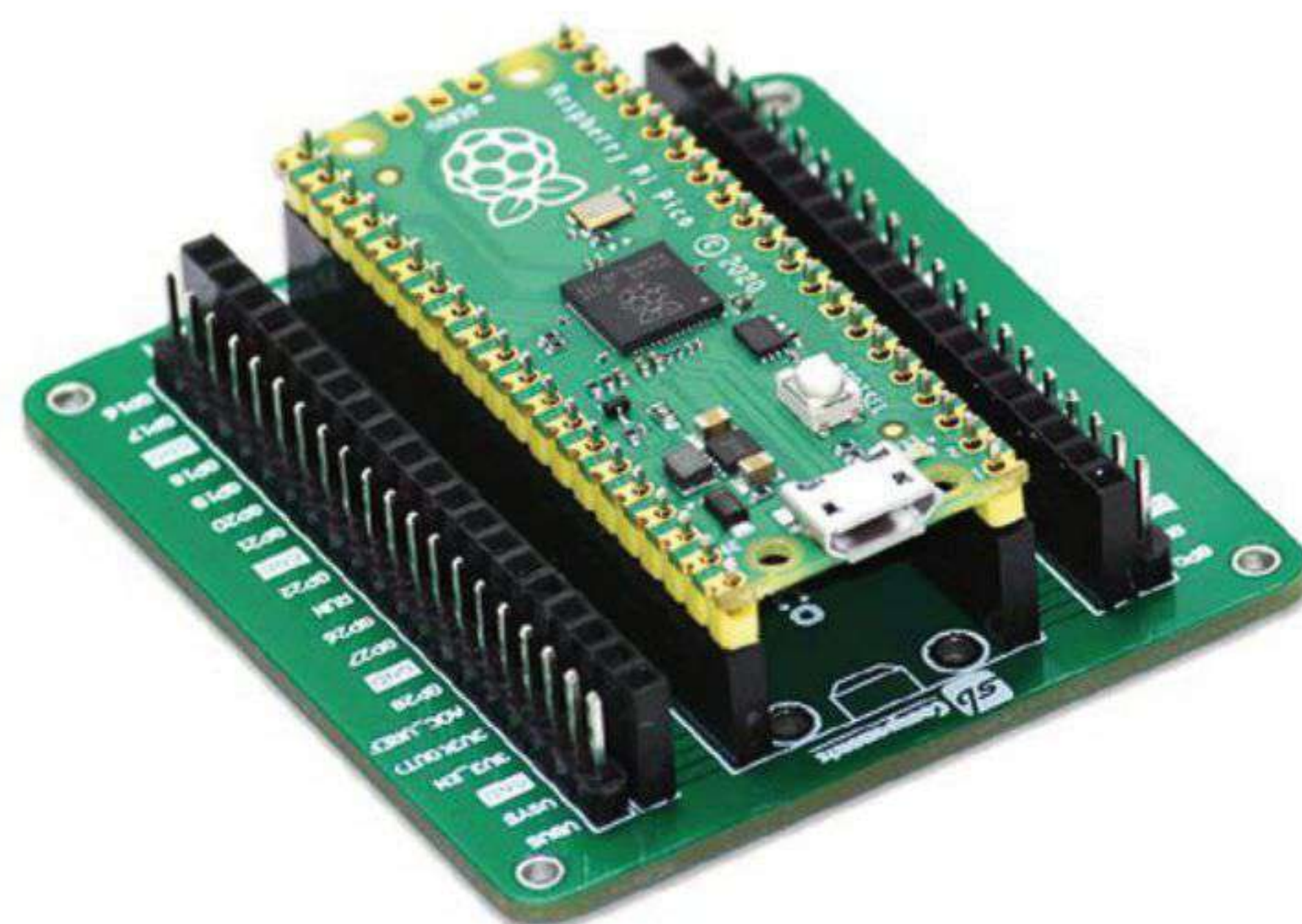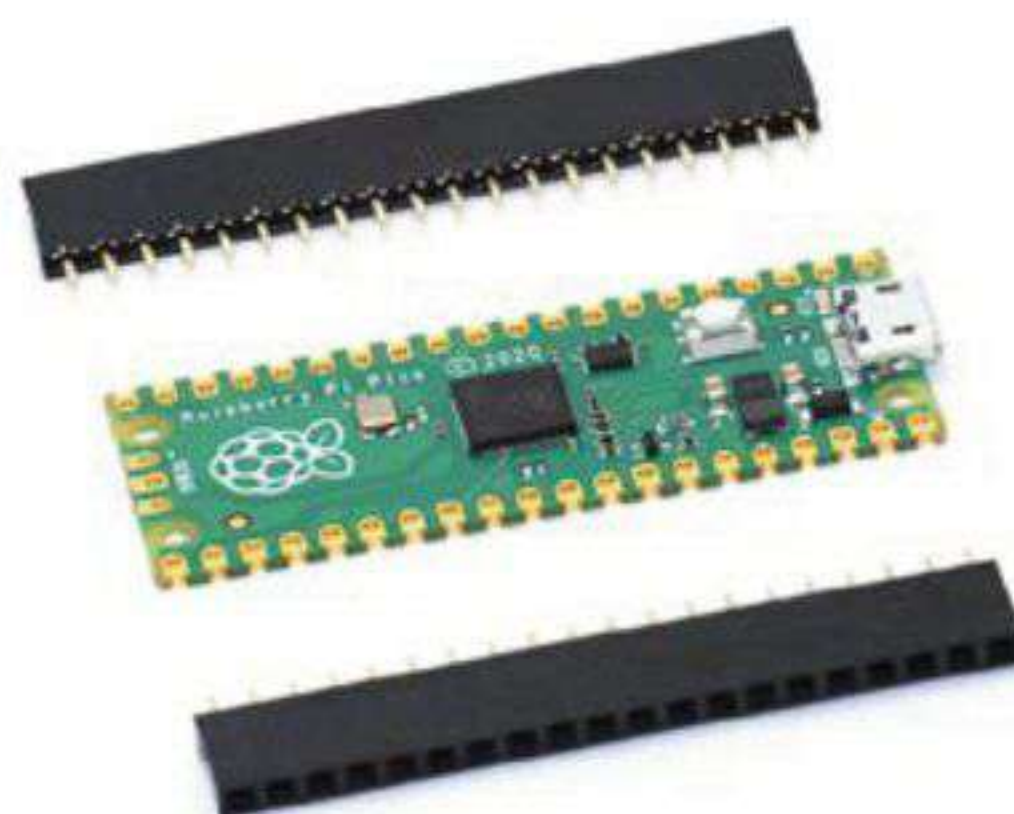
▲ GPIO Expansion Board

Solderless pin access

If you don't fancy soldering headers to your Pico, this board allows you to make use of both male and female headers for whatever project you're working on. It's great for prototyping.

£8 / $11 | **magpi.cc/picogpio**

▲ Audio Pack

Portable stereo sound

Turn Pico into a simple sound card / amplifier for a Raspberry Pi, or other hardware if you know how to tweak it. It wouldn't be too hard to turn this into part of a mini MP3 player as well, in case you miss those.

£14 / $16 | **magpi.cc/audiopack**

▶ Display Pack

Tiny and colourful

This is a 1.14-inch, RGB, IPS LED display, which is ridiculous! Pico can power it fairly easily, and the display itself has some physical buttons on it. You could easily use it for simple games or costume wearables.

£14 / $16 | **magpi.cc/displaypack**

▲ Header Pack

Male headers

While not essential, soldering on headers to a Raspberry Pi Pico makes it easy to add it to a breadboard or use jumper wires to attach to other circuit bits. Time to practise your soldering skills.

£2 / $2 | **magpi.cc/headerpack**

Learn game development
with Raspberry Pi

Get a head start with Raspberry Pi game creation with this resource list. By **Mark Vanstone**

itch.io

For most developers of games, the main reason to create a game is to challenge others to play their game. So the first question is: how can we make a game available for other people to play? That's where itch.io comes in. The website provides an app store-style platform for independent developers to upload and sell their games.

Games can be built and uploaded in all kinds of formats. They can be built as executables, source code downloads, or online browser games. There is a large active community, and regular competitions to reward the best games. The itch.io site is free to use and provides lots of support for new developers and if you want to sell your game, they will deal with all the payment process but, of course, ask for a small cut of the profits.
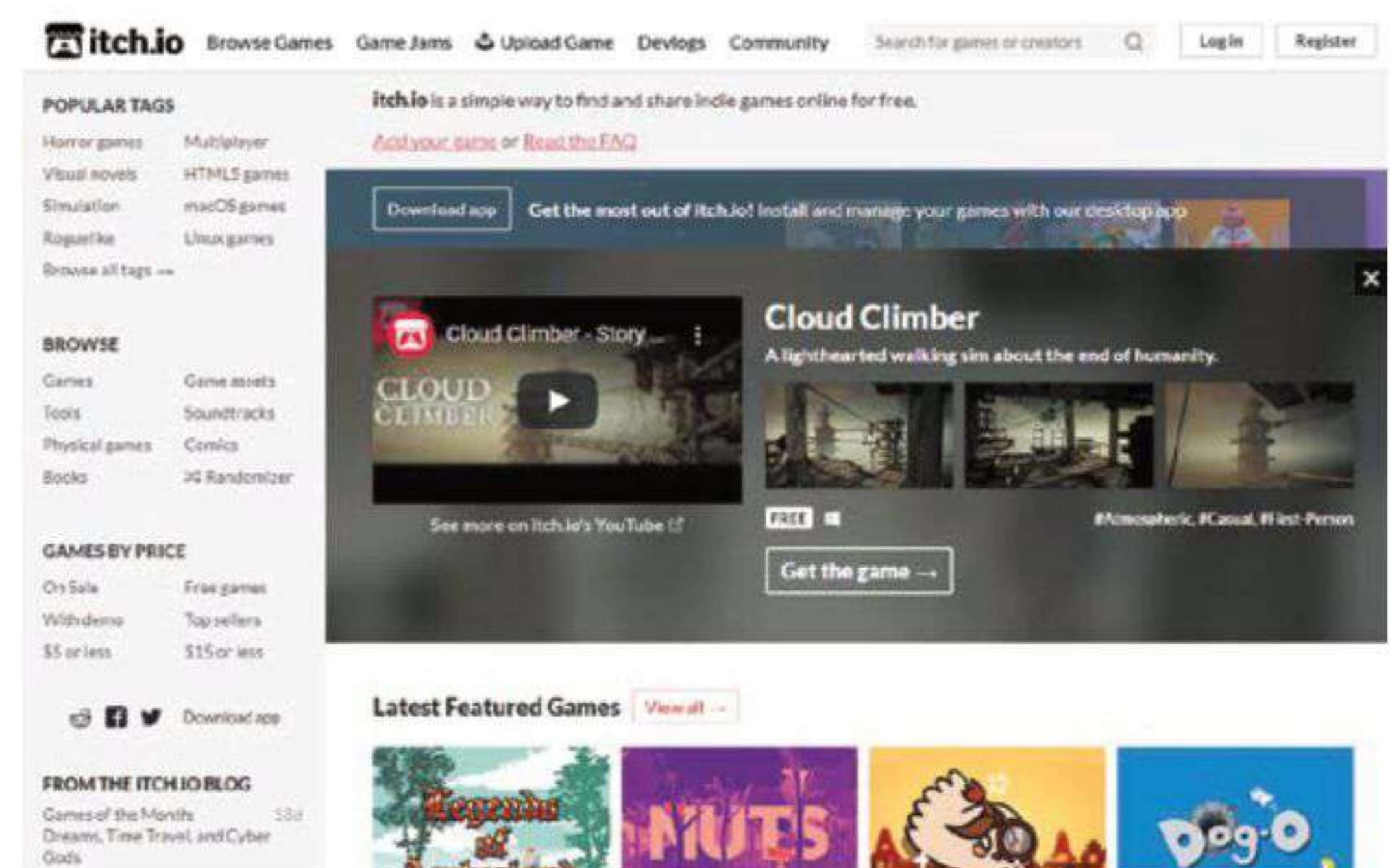
Currently there are over 300,000 games hosted on itch.io, so you can have a good look around and see what everyone else has uploaded, and get ideas about how to present your new game to the world, get feedback from players, and even make a bit of money.

Books for game development

Paper-based or online books for reference and tutorial
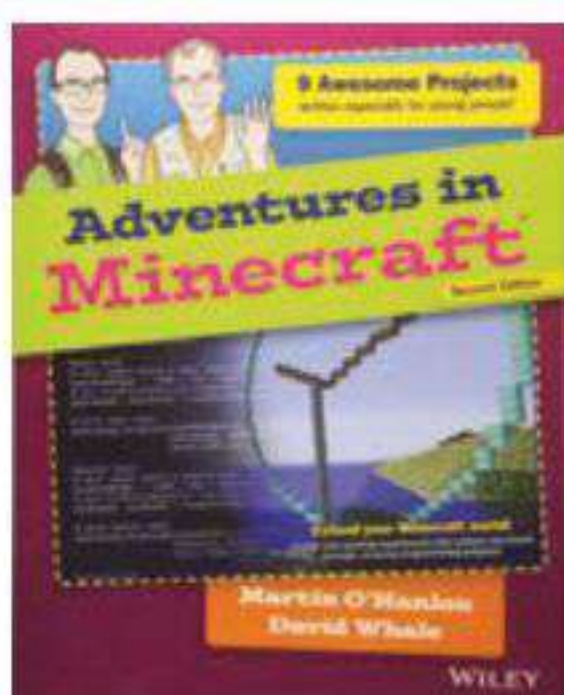
ADVENTURES IN MINECRAFT

A treasure-trove of a book, both paper-based and for download. Learn to build games using the Minecraft engine, and even program external controllers to trigger events in-game.
magpi.cc/advminecraft

CODE THE CLASSICS

Learn how to create your own versions of retro games from scratch using Python and Pygame Zero. Five classic video games are remade, ranging from Pong to Sensible Soccer.
magpi.cc/codetheclassics

MAKE GAMES WITH PYTHON

An in-depth look at game creation with Python and Pygame. From your first game to physics simulations and alien invaders, this book is packed full of useful techniques and listings.
magpi.cc/makegamespython

Scratch

Scratch is available as a game and animation development system, both in a browser and as an offline program. Both work and look very similar. Scratch is an excellent introduction to programming, and provides a visual block interface to create interactive content. You can share games that are created with Scratch, and there are lots of examples on Raspberry Pi's website for you to see what others have done with Scratch. Graphics and sounds are included in the Scratch library, but you can also create your own using the built-in pixel editor or a separate paint package. There are extra extensions you can add to connect to external projects, and a whole range of tutorials to show you how to get started making the game of your choice.

Raspberry Pi Game Projects

Since the launch of Raspberry Pi, the Raspberry Pi Foundation has been producing example projects of all kinds on its website. In the game section, there are around 60 projects for you to delve into and find out how they were made and download the elements you need to build them. There are projects for Scratch, Python, web browsers, and even games to play with external hardware, like the Sense HAT. Each project describes what you will need to make it, shows the finished project, and then walks you through, step by step, what you need to do from beginning to end. You will also find suggestions for other projects to look at after you have finished, to progress further with your game development experience. **M**

Game Creator Resources

Get free game resources online

PI GAME DEV

Pi Game Dev is a well-organised site dedicated solely to resources for making games on Raspberry Pi. There are comprehensive lists and links to game engines, art and music tools, code editors, and game assets.
pigame.dev

OPENGAMEART.ORG

OpenGameArt is a go-to, one-stop, free shop for 2D and 3D game graphics and sound effects. Searchable and categorised, this site features thousands of submissions from designers. You can submit your own creations to give something back.
opengameart.org

SPRITERS RESOURCE

Sprite sheets are bitmaps full of animation frames, and Spriters Resource has all the retro game graphics in sprite sheet format. Just search for your favourite game and there's likely to be a sprite sheet or two for it.
spriters-resource.com

Tanya Fish

Tanya is all over the online Raspberry Pi community.
How did this engineer, teacher, and student start out?

> ➤ Name **Tanya Fish** | ➤ Occupation **Student**
> ➤ Community role **Educator and maker** | ➤ url **@tanurai**

▼ Tanya continued the tradition of making earrings resembling new, tiny Raspberry Pi boards – this one is a handmade Pico

If you're part of the online Raspberry Pi community, it's hard to miss posts from **Tanya Fish, aka tanurai.** She creates cool things and has great insights into making.

"I used to program the BBC Micro using BASIC code from the back of magazines as a kid, and got in trouble at school for altering Granny's Garden (an educational puzzle-solving game)," Tanya tells us. "When I was 16, my dad put a Ford Sierra in front of me and said 'take that apart'. I'd been taking things apart throughout my childhood, and I think that was the first thing that was physically useful. I was lucky enough to work with a drag racing team for a few years on a Fuel Altered [car] that did the quarter mile in 7.4 seconds. I carried on making anything I could – woodworking, drawing, knitting, crochet, smaller things mainly. Then I discovered Raspberry Pi and got into learning my way around a Linux operating system, and started to teach myself Python."

After working for our cool, gadget-making pals Pimoroni, Tanya is back at school working towards a PhD "in the effects of STEM outreach in schools".

When did you learn about Raspberry Pi?
Probably in the later half of 2012, and a lot of my friends were using them as media servers. I got one for Christmas 2012 – and I still have it! I've used every model since, and I liked it that they were small enough to build into projects.

▲ Tanya has also presented at Maker Faire

What is your favourite way to interact with the community?

My favourite way to get involved in the community is by volunteering at events like Raspberry Jams, sometimes giving talks, workshops, or just bringing along a project to talk about. Luckily, my previous employers were really supportive of that, and I loved standing at the stall and chatting with people about what they were making, and helping out with equipment choices. I try to write up personal projects, but documentation is time-consuming!

What has your experience been like with Raspberry Pi?

I think there's a lot more to come from Raspberry Pi. I have never been a computing teacher, yet I have used a Raspberry Pi and coding for every subject I've taught. There's a lot to be said for their creative use – last year one of my graphics students made an interactive video player controlled by children's toys to teach history – and to say that they can go from not knowing any code to that shows the ease of use of Raspberry Pi. I look forward to using them in my teaching for years to come. 𝐌

> " **I think there's a lot more to come from Raspberry Pi** "

Big projects

Eurovision crown

"I think that the first big project I did was with a Raspberry Pi Zero W, for a Eurovision party in 2017. It used APA102 pixels and a Scroll pHAT, and ran off a Poundland battery pack. I was just starting out with Python then, but I managed to make it grab tweets about Eurovision and scroll them across the front of the crown."

Giant Picade

"A project I built a workshop around was a giant breakout of the controls to play games using RetroPie on a Raspberry Pi 3B+. I got kids to build all of the switches using cardboard and foil, and clip them onto this rig. We were playing Mario with a spoon, flip-flops, head-desks, and homemade tilt switches."

Elephant nightlight

"I also did a kids' nightlight using a Raspberry Pi Zero, which changes colour depending on whether it's time to get up or go to sleep. I liked that Raspberry Pi Zero is small enough to fit inside a 1 pint milk bottle! It's actually a useful project; most of the things I build are frivolities – I like playing with tech creatively."

This Month in
Raspberry Pi

MagPi Monday

Amazing projects direct from our Twitter!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

Here's a selection of some of the awesome things we got sent this month – and remember to follow along at the hashtag #MagPiMonday!!

01. MacFeegle Prime is the mini Johnny 5 of our dreams

02. We had a couple of people show us time-lapses over the last month, and they're all great

03. Bread rising is a miracle of food science

04. As you may imagine, it does not work well

05. While this may look like Asteroids, it's just a very pretty display

06. We love seeing this project evolve!

07. An amazing build here, worthy of the space station

08. This robot clearly means business

09. This *Star Trek TNG* tricorder looks extremely legit

10. Snow is very cool. It's fun to watch it build up like this

Keegan Neave
@NeaveEng

01

Replying to @TheMagPi

More work on my #piwars robot #MacFeeglePrime! Just got both cameras working and sending video to my custom controller :)

Running on a @StereoPi board, slowly coming together!

MB
@MB76978199

02

Replying to @TheMagPi

a time lapse for germinating seeds

maxi14x
@maxi14x

03

Replying to @TheMagPi

Just watching bread rise

8 Bits and a Byte
@8BitsandaByte

04

Replying to @TheMagPi

We built an AI Label Maker to make home organisation more efficient! 🤖

Spoiler alert: it works about as well as you imagine ✨

Dr Footleg - Roboteer
@drfootleg

05

Replying to @TheMagPi

I ran 73,728 LEDs at once off a Raspberry Pi 4, testing my new Pi power PCB design. This iteration could handle 4A continuous load without overheating.

Dr Footleg - Roboteer @drfootleg · Jan 31
Let's go LED crazy!

0:16   3.8K views

11:17 AM · Feb 1, 2021 · Twitter Web App

Rob
@delineator

06

Replying to @TheMagPi

#MagPiMonday Writing scripts to generate microtonal melodies on my monome norns shield / raspberry pi 3b+

Rob @delineator · Jan 31
#january #microtonal #xenharmonic #ElectronicMusic #monome #noms #refaceCP #january2021 5D#7.5 Gorgo-6

Watch again

0:00   1.2K views

1:06 PM · Feb 8, 2021 · Twitter for Android

Tim Zallmann
@tomtim

07

Replying to @TheMagPi

Hopefully the Final stretch for getting my sons spaceship cockpit done!

directive0
@directive0

09

Replying to @TheMagPi

I'm making a Raspberry Pi based Tricorder.

6:38 PM · Jan 25, 2021 · Twitter Web App

Laurence Molloy
@MolloyLaurence

08

Replying to @TheMagPi

I spent this weekend helping my eldest son (15) polish off his Arkwright Engineering Scholarship application - his reference project is his involvement in a PiWars team this year, solving the Tidy Up The Toys challenge.

Here are some photos of his early robot attachment designs.

Laser Diode   Distance Sensor

Andrew Evans
@AndrewEvans0102

10

Replying to @TheMagPi

I did a time lapse of a snow storm we just had using Pi Zero attached to a USB webcam. Added some music and even put it on YouTube. 😊📹

Snow Time Lapse Richmond Va January 2021 (fast)
This is a time lapse of snowfall we just had in the Short Pump area of Richmond. I took it with my Raspberry Pi over the ...
🔗 youtube.com

11:39 PM · Feb 1, 2021 · Twitter for iPhone

Mondays are for Pico

#MagPiMondays aren't just for standard Raspberry Pi projects

Raspberry Pi Pico was released two days after we went to press last issue, so we weren't able to feature any community projects then.

This month, though, we have plenty. Take a peep at some Pico projects that were shared with us on #MagPiMonday. 🅜

Kevin McAleer @kevsmac
Replying to @TheMagPi
01
Adding a Pico to my opencat!
1:36 PM · Jan 25, 2021 · Twitter for iPhone

Mike Cook @Wee_Grumphie
Replying to @TheMagPi
02
Generating tones from the PIO state machine toggling a pin:-

Mike Cook @Wee_Grumphie · Feb 8
Been playing about with note generation, using the PIO state machine, on the #pico got this simple tune going in microPython. Does any one know how to set up or change a state machine from inside a function? @Raspberry_Pi

0:21 949 views
1:14 PM · Feb 8, 2021 · Twitter Web App

Anne Carlill @Yorks_STEAM
Replying to @TheMagPi
03
One large step for me but a small step for #RaspberryPiPico . Got the LED to light up on the Pico #MagPiMonday
3:16 PM · Jan 25, 2021 · Twitter for iPhone

Tanya Fish @tanurai
Replying to @TheMagPi
04
Putting coloured headers on pico number 2. Yes, I realise a few of my choices are unconventional.
11:06 AM · Feb 8, 2021 · Twitter for Android

Brian Cortell @CannonFodder
Replying to @TheMagPi
05
My Tom, who is 11 soldered up his Raspberry Pi Pico* then flashed, dimmed LEDs, controlled them with a button and a pot.
Other news: I just made my Pico flash a LED and started programming a web interface to control a #MicroPiNoon robot 🤖
*Very pro job on the soldering!

Caroline Dunn @thecarolinedunn
Replying to @TheMagPi
06
I worked on a #raspberrypi Pico Hydrometer with LCD screen for @tomshardware #raspberrypipico in C/C++ using the Pico-SDK

01. Would a Pico Cat be a Picat?

02. *The MagPi* regular Mike Cook has a go with music on Pico using PIO

03. A very important thing to learn!

04. Colour-coding pins is a great way to make prototyping a little quicker

05. We could not solder that well at eleven years old

06. As usual, Caroline is getting stuck in doing cool stuff with Pico

Best of the rest!

Other amazing things from the community

RASPBERRY PI 4 DJ SETUP

This setup looks amazing, and Reddit user Error_No_Entity has included an OS image so you too can have this excellent build. We'd love to know more about the physical build, though.

▶ **magpi.cc/djpi4**

DRIVING OBJECT DETECTION

This is an incredible build using Raspberry Pi 4, TensorFlow, and publicly available training datasets. It's amazing what you can do with computer vision on a Raspberry Pi.

▶ **magpi.cc/rpiroad**

RETRO_MATRIX
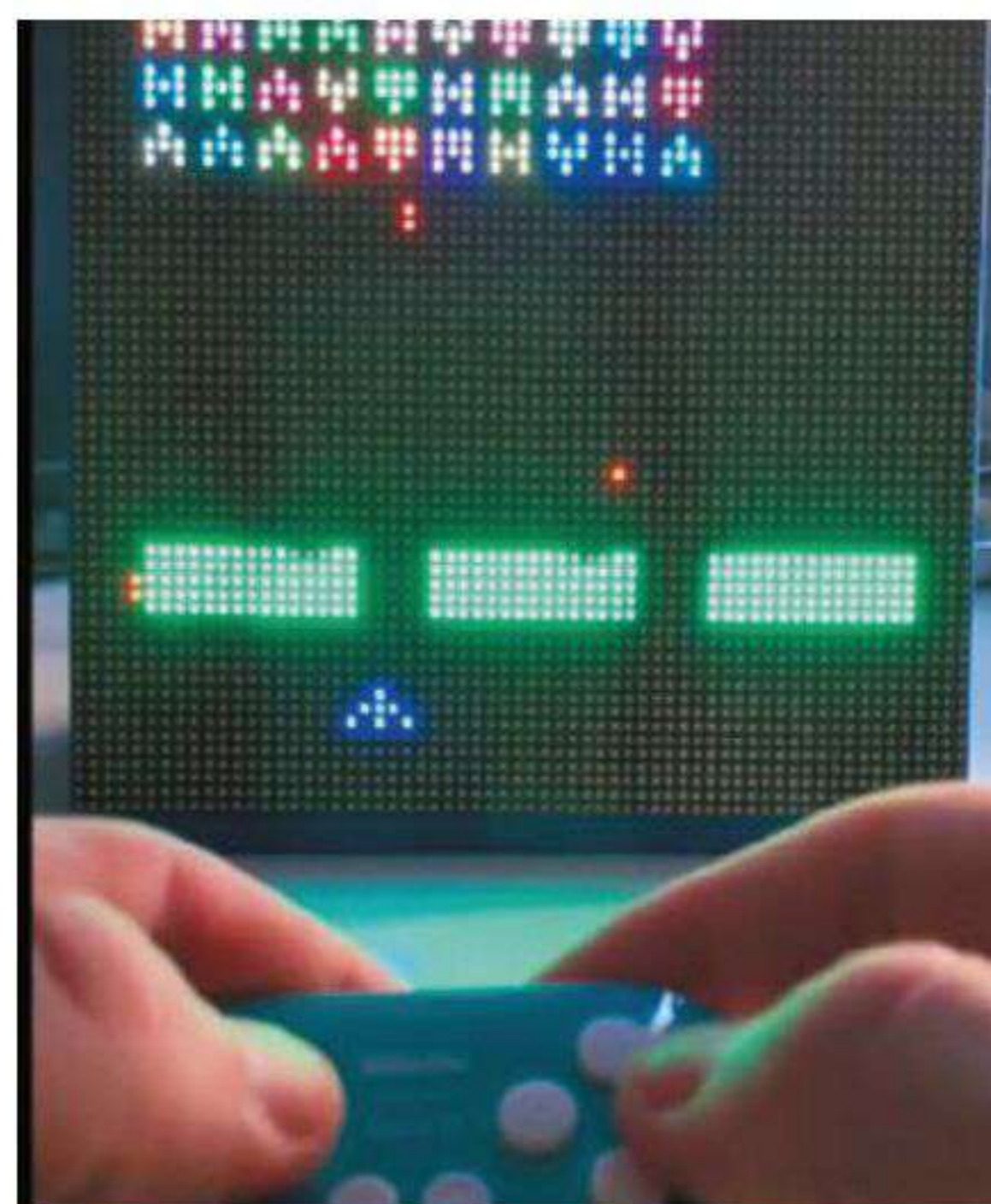
Reddit user Gorse212 has been customising retro games to work on a large LED matrix, and they look pretty cool! This could easily be scaled up to larger projects as well, we think.

▶ **magpi.cc/retromatrix**

ANOTHER AIRPORT RASPBERRY PI

"My local rinky-dink airport apparently runs the arrivals tracking on a [Raspberry Pi 4]," points out V0xx0m on Reddit. It seems like more and more airports are using them. Hopefully they'll fix this one.

▶ **magpi.cc/pi4airport**

PLOTTYBOT
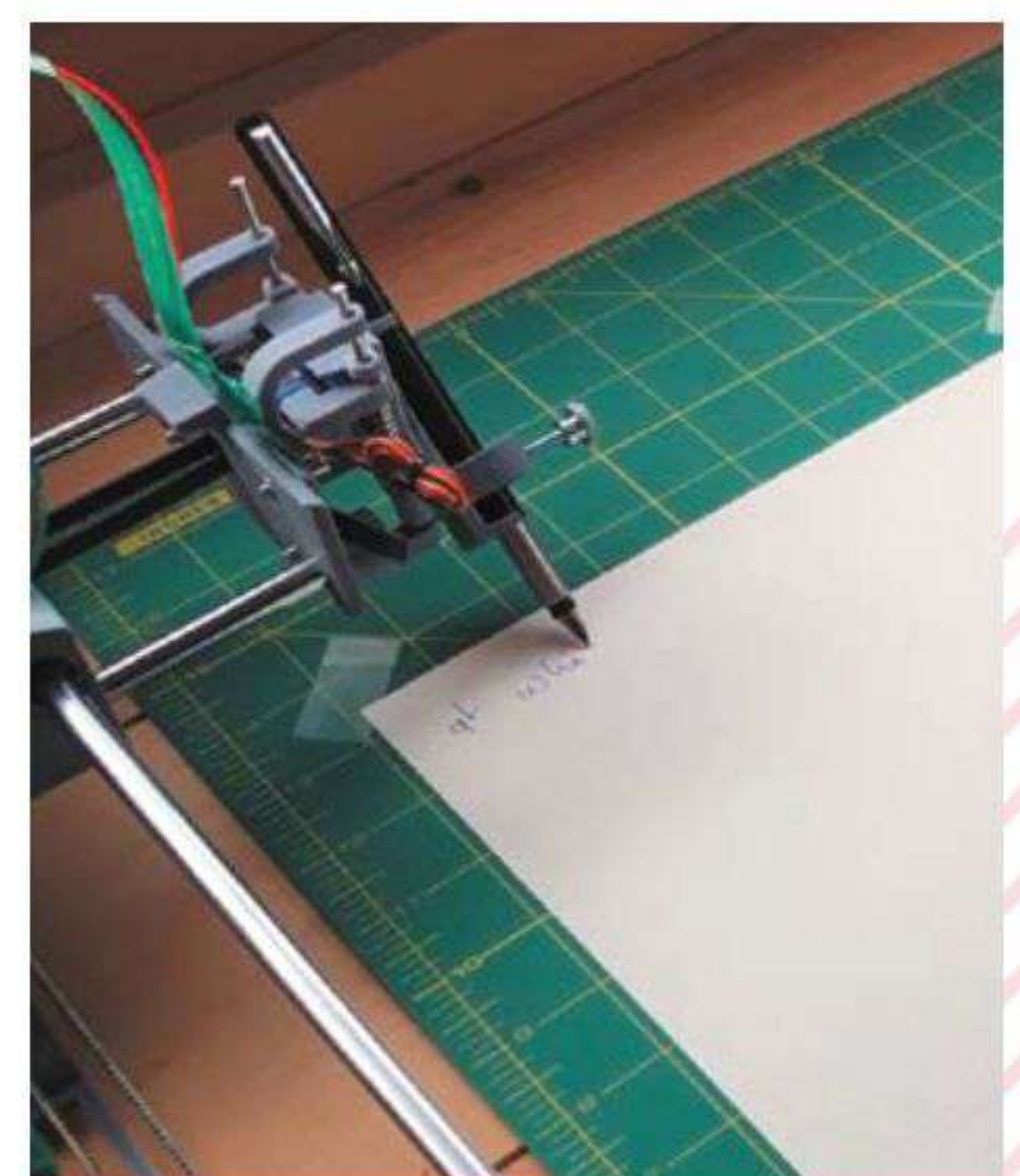
This ingenious build perfectly copies not only your handwriting, but will also write out text that you've typed. Now handwritten notes don't have to be… handwritten.

▶ **magpi.cc/plottybot**

Your
Letters

RP2040 minus Pico

Will the RP2040 be available to buy as an IC from usual suppliers? I really want to use it as the base IC for a new sensor board, but the footprint of the Pico module is too big.

Jah via email

According to our friends at HackSpace magazine: "It will be once there's enough of them created. We're expecting Q2 this year, but this isn't confirmed yet."

For those not in the know, you can buy and use RP2040, the core chip that powers Raspberry Pi Pico. A few microcontroller makers are already coming out with their own ranges of boards with RP2040 chips, and the sky's the limit.

▲ However you like to use your Pico is up to you
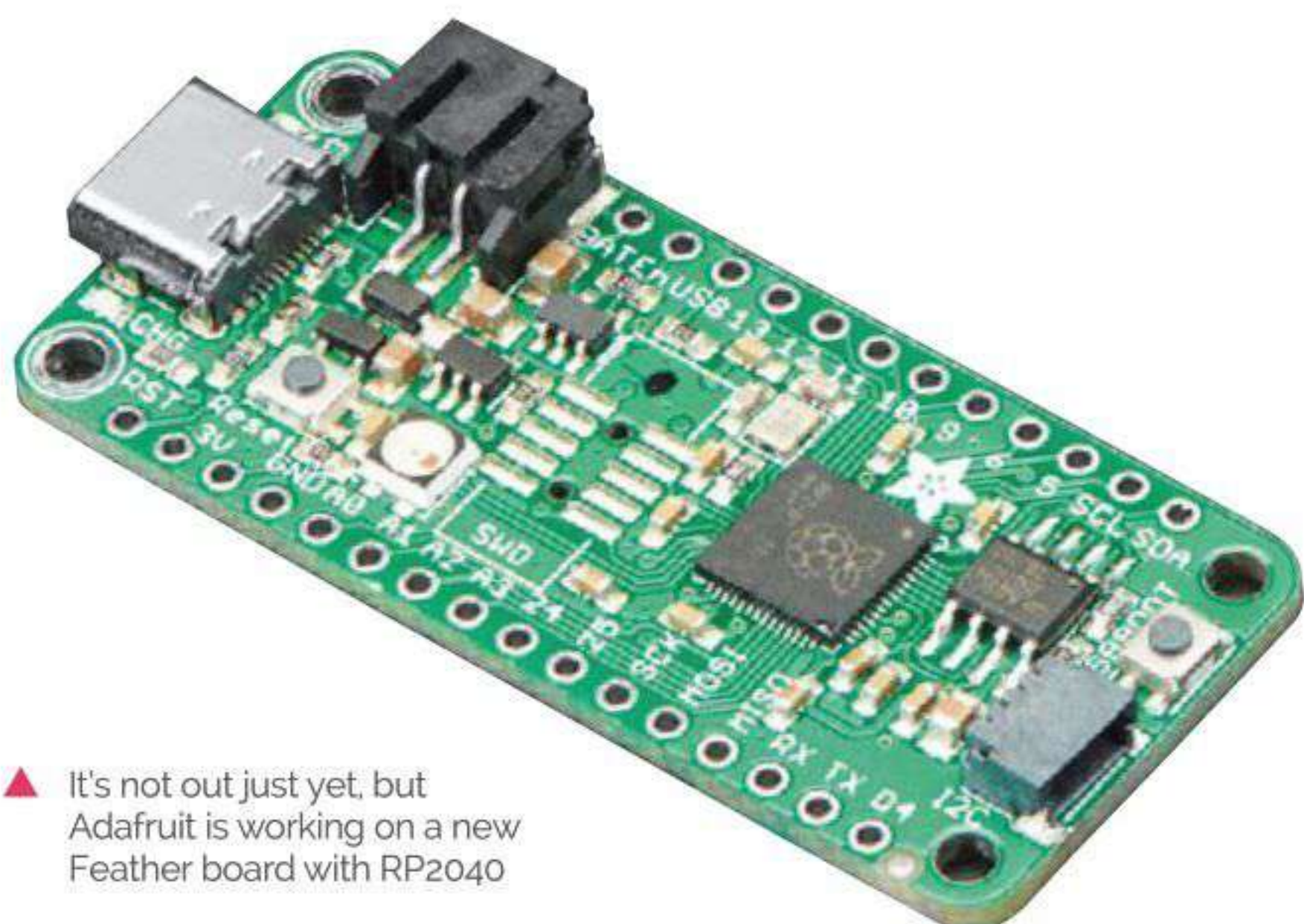
▲ It's not out just yet, but Adafruit is working on a new Feather board with RP2040

Pre-soldered Pico

Is there any way to get a Raspberry Pi Pico that has the pins already soldered onto it? My soldering skills aren't quite up to scratch, I'm afraid.

Charlotte via Facebook

You can actually get pre-soldered Pico boards from many of the Pico resellers, such as SB Components and Pimoroni. They're usually in more limited supply, though, and will cost a bit extra.

We personally like the holes, as for some wearable stuff it's a bit more useful. Also, it would be good practice to solder on the headers anyway!
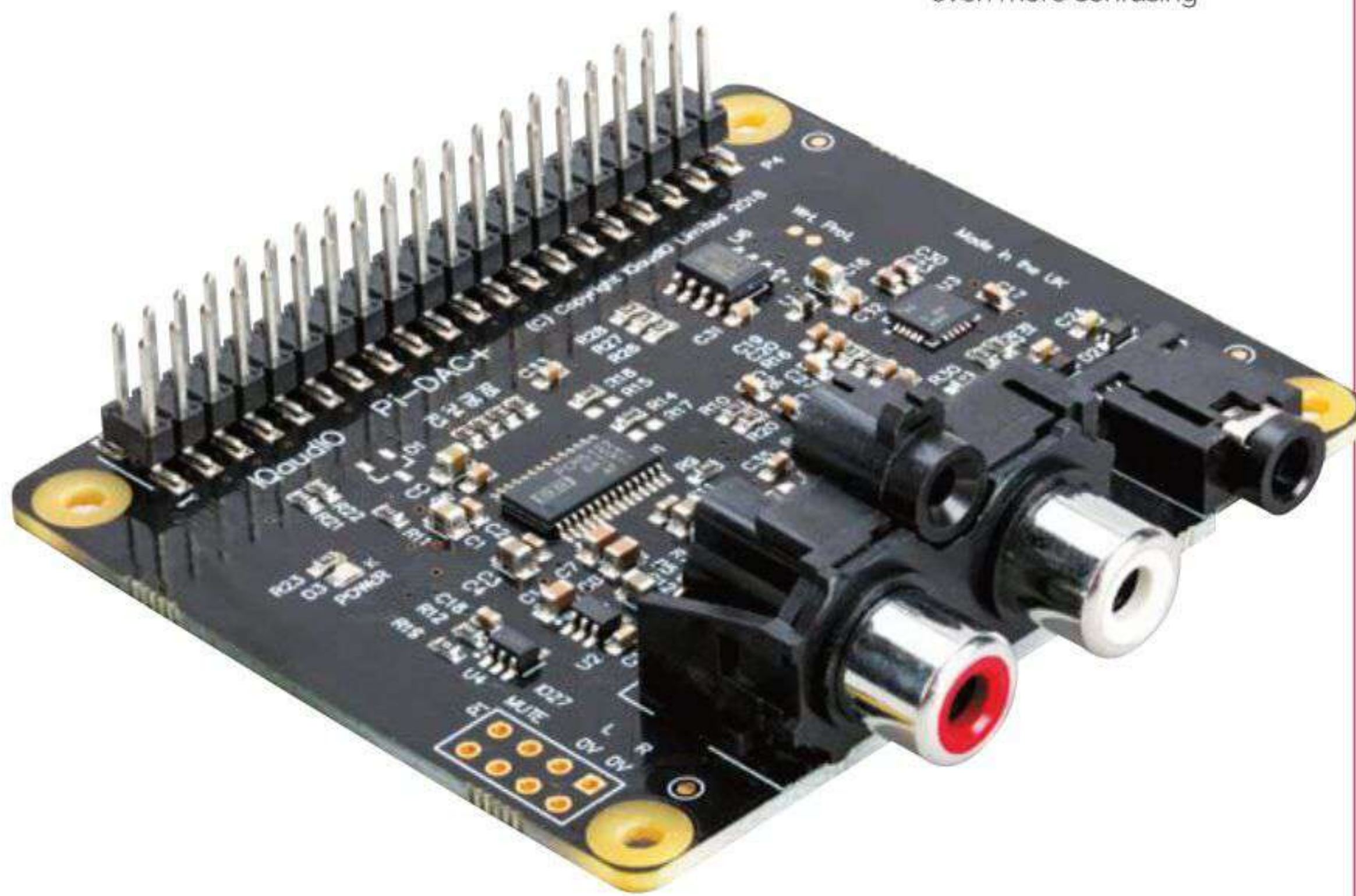
This DAC is also a HAT, which we understand makes things even more confusing

Coolest Projects

When does Coolest Projects registration open? Is it separated by locale like in previous competitions? I'd really like something to work towards right now!

Dale via twitter

You're in luck, as registration has recently opened! You can register here: **magpi.cc/cpregister**.

This year the competition is worldwide, and everyone up to the age of 18 can participate! There are loads of categories you can apply for, and there are workshops and tips on how to manage your project. You have until 3 May to apply, so get online and get started!

Glossary

Any chance of an up-front glossary on your website for complete novices like me? There are so many acronyms – HAT, pHAT, DAC, etc. – that I get completely lost in having to look them up elsewhere. And even then, they are not always well explained because they assume another level of knowledge.

Perhaps there is one, but I couldn't find it!

Nick via email

Submit your project to be in the online showcase gallery!

You're correct, we don't currently have a glossary of terms like this. However, it's a good idea, and we'll try and get one set up. We'll put it on GitHub so folks can make suggestions for it – otherwise, any acronyms or terms you think of, don't hesitate to let us know so we can add them!

Contact us!

> Twitter **@TheMagPi**
> Facebook **magpi.cc/facebook**
> Email **magpi@raspberrypi.com**
> Online **raspberrypi.org/forums**

WIN ONE OF TEN RFID HATS!

The RFID HAT for Raspberry Pi allows you make use of RFID tags that you already have, or the programmable card or fob that comes with it. It even has a little LCD screen. We gave it an 8/10 last issue, and now ten are up for grabs.
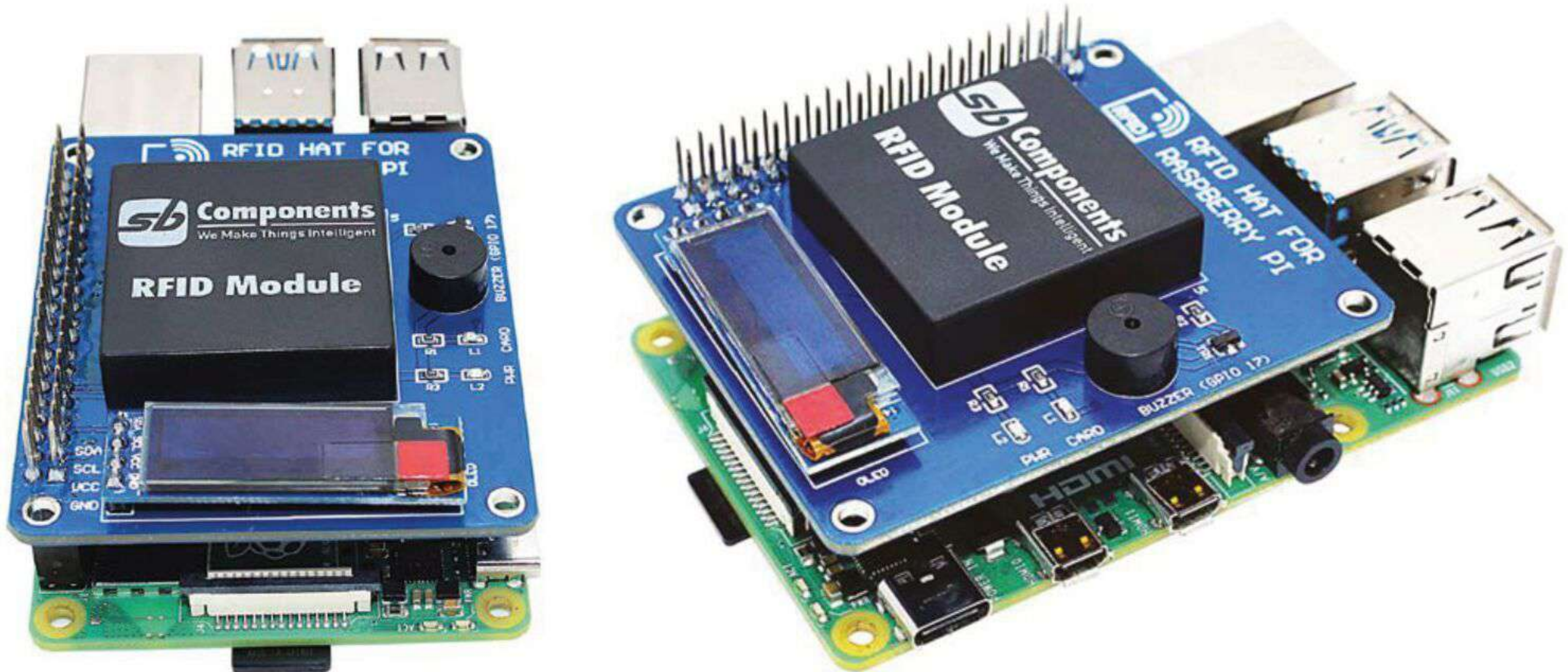
Learn more: magpi.cc/rfidhat | **Head here to enter: magpi.cc/win**

Terms & Conditions

Competition opens on **24 February 2021** and closes on **25 March 2021**. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

HOME AUTOMATION
WITH RASPBERRY PI
AND PICO

MAKE YOUR HOME SMARTER WITH RASPBERRY PI COMPUTERS AND MICROCONTROLLERS
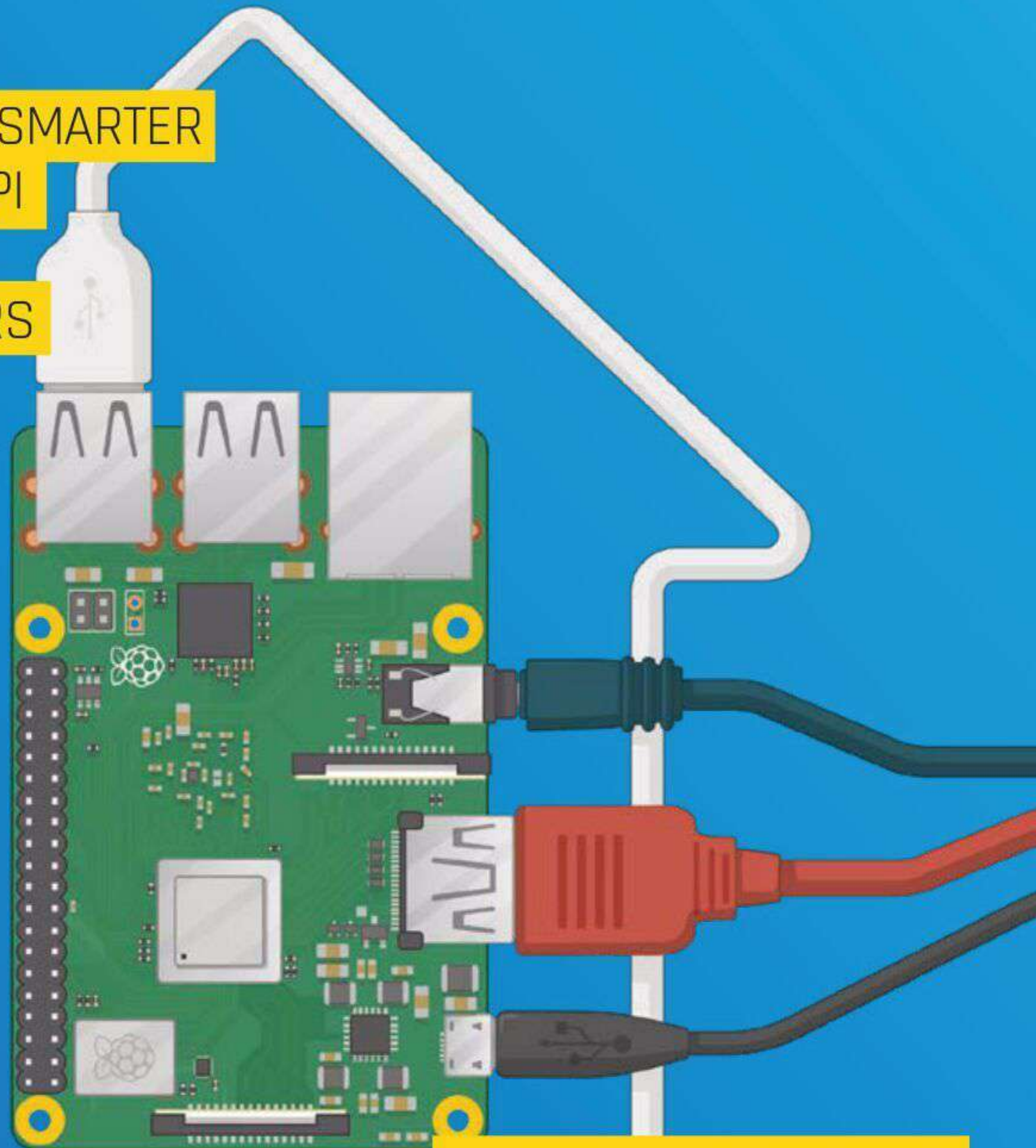
THE MAGPI #104 ON SALE 25 MARCH

Plus!

Learn to code with Raspberry Pi

Start building an arcade machine

Discover good GUI design skills

DON'T MISS OUT!
magpi.cc/subscribe

TWITTER	@TheMagPi
FACEBOOK	fb.com/MagPiMagazine
EMAIL	magpi@raspberrypi.com

Vertical innovation

James Adams on building Raspberry Pi's first microcontroller platform

O n 21 January we launched the $4 Raspberry Pi Pico. As I write, we've taken orders for nearly a million units, and are working hard to ramp production of both the Pico board itself and the chip that powers it, Raspberry Pi RP2040.

Microcontrollers are a huge yet largely unseen part of our modern lives. They are the hidden computers running most home appliances, gadgets, and toys. Pico and RP2040 were born of our desire to do for microcontrollers what we had done for computing with the larger

well as requiring specialist skills, you need a lot of expensive tools and IP. After a slow start, for the past couple of years we've had a small team working on it full-time, with many others pulled in to help as needed.

Low-cost and flexible

The Pico board was designed alongside RP2040 – in fact we designed the RP2040 pinout to work well on Pico, so we could use an inexpensive two-layer PCB, without compromising on the layout. A lot of thought has gone into making it as

can be programmed to 'bit-bang' almost any digital interface without using valuable CPU cycles. Finally, we have released a polished C/C++ SDK, comprehensive documentation and some very cool demos.

For me, this project has been particularly special as I began my career at a small chip-design startup. This was a chance to start from a clean sheet and design silicon the way we wanted to, and to talk about how and why we've done it, and how it works.

Pico is also our most vertically integrated product; meaning we control everything from the chip through to finished boards. This 'full stack' design approach has allowed optimisation across the different parts, creating a more cost-effective and coherent whole.

And of course, it is designed here in Cambridge, birthplace of so many chip companies and computing pioneers. We're very pleased to be continuing the Silicon Fen tradition.

> " This 'full stack' design approach has allowed optimisation across the different parts "

Raspberry Pi boards. We wanted to create an innovative yet radically low-cost platform that was easy to use, powerful, yet flexible.

It became obvious that to stand out from the crowd of existing products in this space and to hit our cost and performance goals, we would need to build our own chip. I and many of the Raspberry Pi engineering team have been involved in chip design in past lives, yet it took a long time to build a functional chip team from scratch. As

low-cost and flexible as possible – from the power circuitry to packaging the units on to tape and reel (cost-effective with good packing density).
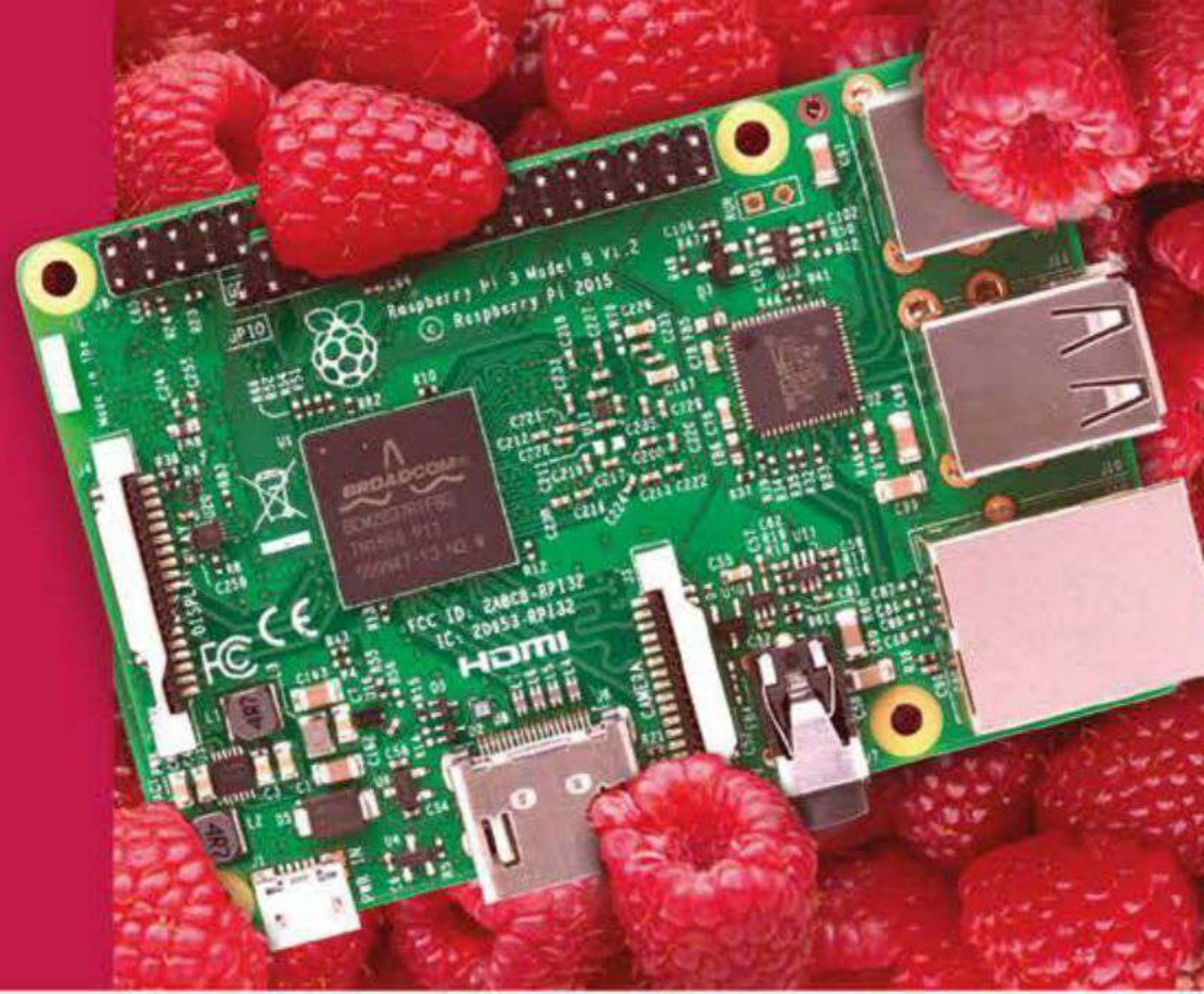
With Pico we've hit the 'pocket money' price point, yet in RP2040 we've managed to pack in enough CPU performance and RAM to run more heavyweight applications such as MicroPython, and AI workloads like TinyML. We've also added genuinely new and innovative features such as the Programmable I/O (PIO), which

AUTHOR

James Adams

As Raspberry Pi Trading's Chief Operating Officer and Hardware Lead, James has been deeply involved in Raspberry Pi product development since 2013.

@JamesAdams314